ROBOT LEARNERS: INTERACTIVE INSTANCE-BASED LEARNING WITH SOCIAL ROBOTS

A Thesis
Presented to
The Academic Faculty

by

Hae Won Park

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Electrical and Computer Engineering

Georgia Institute of Technology
May 2014

ROBOT LEARNERS: INTERACTIVE INSTANCE-BASED LEARNING WITH SOCIAL ROBOTS

Approved by:

Professor Ayanna M. Howard, Advisor
School of Electrical and Computer Engineering
Georgia Institute of Technology

Professor Andrea L. Thomaz
School of Interactive Computing
Georgia Institute of Technology

Professor Henrik I. Christensen
School of Interactive Computing
Georgia Institute of Technology

Professor Linda M. Wills
School of Electrical and Computer Engineering
Georgia Institute of Technology

Professor Charles C. Kemp
Department of Biomedical Engineering
Georgia Institute of Technology

Date Approved: 31 March 2014

ACKNOWLEDGEMENTS

First and foremost, I would like to thank my advisor, Prof. Ayanna Howard. We have went through tremendous amount of adventures together, and I am excited about what lies ahead of us. Thank you for teaching me the most valuable aspects of robotics and engineering - making a social impact and improving people's quality of life. Thank you for exposing me to people with various disabilities and their unique needs that provided the inspirations for this dissertation topic. Your unwavering passion toward research has shaped the researcher that I am today.

My thesis committee provided me with the best experience at Georgia Tech. After many years of admiring their work, I was thrilled when they agreed to become my committee members. Thank you professors Henrik Christensen, Andrea Thomaz, Charles Kemp, and Linda Wills. I cannot thank you enough for the encouragements and advices you gave me toward my dissertation and career path, and for appreciating my work and giving me the strength to carry on this dissertation topic. I'd also like to thank professors Paul Oh, Dennis Hong, Daniel Lee, and Aaron Lanterman, for introducing me to new opportunities and reminding me just how fun research is.

To my academic brothers and sisters, a.k.a. guinea pigs, of the HumAnS lab: Sekou Remy, Antidio Viguria, Stephen Williams, Douglas Brooks, Chung Hyuk Park, Lonnie Parker, Jiuguang Wang, Gregorio Drayer, Richard Coogle, Paul Robinette, Sergio Garcia, LaVonda Brown, Kevin DeMarco, Mason Nixon, Nicole Giullian, Giancarlo Valentin, and Brittney English. I thank you for the days and nights we spent together discussing research ideas and preparing robot demos. I will miss our potlucks, barbecue parties, and trips to various conference locations. I apologize for slowly colonizing every available desk space, which I am pretty sure you all had fun

watching. My robotics friends at Georgia Tech: Crystal Chao, Alex Trevor, Tiffany Chen, Changhyun Choi, Kimoon Lee, and Carlos Nieto, thanks for your years of friendship and for going through this PhD process together. I hope to see you at future conferences!

I wouldn't have loved this city as much without these people who became my extended family: Seonghye Jeon, Judy Lee, Milim Lee, Hinkyeol Woo, Lynn Huh, Moojoong Ra, Donggu Choi, and Chuljin Park. Your presence throughout my grad-student life has made this walk all the more bearable.

To my parents and brothers, I wouldn't have made this far without your consistent encouragement and love. Thank you for enjoying my often awkward inventions since childhood and believing in my potentials. Moon Cheon Kim and Jong Choon Park, mom and dad, you have always been my source of energy and ambition. Lastly, to my husband and best friend, Jeong Hoon, besides being a supportive and loving husband, you always pushed me to take the extra mile when it came to my research. I am grateful for your patience and support throughout this process, and I look forward to fulfilling our life goals together. Muah!

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

On one hand, academic and industrial researchers have been developing and deploying robots that are used as educational tutors, mediators, and motivational tools. On the other hand, an increasing amount of interest has been placed on non-expert users being able to program robots intuitively, which has led to promising research efforts in the fields of machine learning and human-robot interaction. This dissertation focuses on bridging the gap between the two subfields of robotics to provide personalized experience for the users during educational, entertainment, and therapeutic sessions with social robots. In order to make the interaction continuously engaging, the workspace shared between the user and the robot should provide personalized contexts for interaction while the robot learns to participate in new tasks that arise.

This dissertation aims to solve the task-learning problem using an instance-based framework that stores human demonstrations as task instances. These instances are retrieved when confronted with a similar task in which the system generates predictions of task behaviors based on prior solutions. The main issues associated with the instance-based approach, i.e., knowledge encoding and acquisition, are addressed in this dissertation research using interactive methods of machine learning. This approach, further referred to as interactive instance-based learning (IIBL), utilizes the keywords people use to convey task knowledge to others to formulate task instances. The key features suggested by the human teacher are extracted during the demonstrations of the task. Regression approaches have been developed in this dissertation to model similarities between cases for instance retrieval including multivariate linear regression and sensitivity analysis using neural networks. The learning performance

of the IIBL methods were then evaluated while participants engaged in various block stacking and inserting scenarios and tasks on a touchscreen tablet with a humanoid robot Darwin.

In regard to end-users programming robots, the main benefit of the IIBL framework is that the approach fully utilizes the explanatory behavior of the instance-based method which makes the learning process transparent to the human teacher. Such an environment not only encourages the user to produce better demonstrations, but also prompts the user to intervene at the moment a new instance is needed. It was shown through user studies that participants naturally adapt their teaching behavior to the robot learner's progress and adjust the timing and the number of demonstrations. It was also observed that the human-robot teaching and learning scenarios facilitate the emergence of various social behaviors from participants. Encouraging social interaction is often an objective of the task especially with children with cognitive disabilities, and a pilot study with children with autism spectrum disorder revealed promising results comparable to the typically developing group.

Finally, this dissertation investigated the necessity of renewable context for prolonged interaction with robot companions. Providing personalized tasks that match each individual's preferences and developmental stages enhances the quality of the user experience with robot learners. Confronted with the limitations of the physical workspace, this research proposes utilizing commercially available touchscreen smart devices as a shared platform for engaging the user in educational, entertainment, and therapeutic tasks with the robot learners.

To summarize, this dissertation attempts to defend the thesis statement that a robot learner that utilizes an IIBL approach improves the performance and efficiency of general task learning, and when combined with the state-of-the-art mobile technology that provides personalized context for interaction, enhances the user's experience for prolonged engagement of the task.

CHAPTER I

INTRODUCTION

The dream of one day owning a robot companion, with the capacity to conduct everyday tasks customized to an individual's preferences, motivated this research. We see advancements in robot technology everyday, but the realization of personal robots still seems far away. One of the primary factors that contribute to robots not living up to their expectations is the complexity associated with programming robots. Programming a robot to perform new tasks requires training that is beyond the skill level of most individuals. For robots to become true *personal companions*, non-expert users should be able to program and customize their robot's skills or teach new ones intuitively.

This dissertation focuses on implementing an interactive instance-based learning (IIBL) method for enabling robot learners. With IIBL methods, robots accumulate experience from teaching and learning activities with humans. Using an instance-based approach as an overarching framework for learning, "snapshots" of experience instances are stored in memory formulated as task state and action pairs. These instances provide guidelines to solve similar problems in the future, mimicking the cognitive process of problem solving in humans (e.g., prototype theory [81, 102]).

The tasks that are presented in this dissertation places the robot learner with a human teacher in a shared workspace. This setting allows the teacher to use his/her intuition to endow the robot with knowledge of task features that represent keywords people use to convey task knowledge to others. The robot uses these keywords as conditions for instance extraction and knowledge acquisition during a teacher's demonstration of the task. Aligned with interactive machine-learning methods, such

a setting encourages the teacher to closely monitor and evaluate the robot's behavior, and provide new instances at the moment learning is happening.

IIBL requires no expert skills to train the robot or perform a task other than sharing a small amount of intuition. This *seed* intuition and demonstrated instances trigger the learning, thereby increasing the efficiency and the performance of the robot learner. We use touchscreen tablets that act as the medium through which a person's task demonstration is quantized into a form the robot can interpret. Tablets can facilitate natural transition into a shared workspace since the users are already familiar with the device. Adopting the concept of *learning by teaching* [78], we have placed children and adults in the teacher's position to tutor the robot. Studies have been conducted to analyze social behaviors projected toward the robot learner during teaching activities and to measure the shift in the length of engagement.

1.1 Motivation

1.1.1 Interactive Instance-based Approach to Robot Learning

The human teacher's understanding of the robot's rationale behind its behavior facilitates the development of better robot learners. From the user's perspective, a robot's typical learning process is a black box. The robot takes some form of input, such as demonstrations of a task from the user, and outputs a behavior that is difficult to trace back to its source of reasoning. One might question why it would be important for the user to understand the logic behind a robot's learning process. In humans, learning and teaching are a bidirectional process. The learning result becomes a metric for defining whether the teaching was successful or not, which triggers the change in the teacher's behavior. For this reason, we introduce an instance-based approach to robot learning. Unlike other machine-learning methods that discard training data after generalization, instance-based methods such as case-based reasoning store the

original form of training instances in memory [64]. When the robot generates a behavior that was not expected, instance-based methods can replace or update the instances that caused these results and re-train the system when necessary. In an interactive learning setting in which learning from demonstration (LfD) is utilized, knowing which demonstrations led to which robot behavior can provide persuasive information for explaining the robot's decision. This encourages the human teacher to provide supplemental or more accurate examples when necessary.

LfD and the ability to retrieve and reuse prior experience are both important in building a robot capable of accumulating skills to serve different people's needs. This dissertation proposes to combine interactive machine learning and instance-based methods as a framework for acquiring, encoding, retrieving, and maintaining task experiences. The virtue, and perhaps a disadvantage, of instance-based methods is in that it allows us to make assumptions and predictions based on what worked in the past without requiring complete understanding. Through interactive learning, we attempt to better understand the task by benefitting from human teacher's intuition to encode task features and reduce uncertainty. To increase the stability of the system, we refrain from using common k-nearest neighbors classifiers and propose to use sensitivity measures with neural networks for solving regression problems of computing task-feature contributions.

1.1.2 Robot Learners and a Shared Workspace

Though the application areas of robot learners are endless, the first motivation to build robot learners originated from interactions with children who required special education and entertainment options. Socially assistive robots (SAR) [38] provide assistance through social interactions, and research efforts have been made to address the needs of people with disabilities through SAR platforms. Local clinicians have shown interest in using a robotic platform to motivate patients during therapeutic

tasks, and by placing robots in these environments, we were able to create more attentive and engaging sessions (Figure 1).

(a) Presenting the robot learner to local clinicians.

(b) A boy expressing affection to one of our robots during a session.

Figure 1: Socially assistive robots are gauging attention as therapeutic mediators that motivate individuals to better engage in tasks.

The tasks conducted during a clinical session are customized for each person's unique abilities and developmental stages, and a single task is performed in different ways. Hence, for robots to become effective mediators and motivators, an issue of adaptation has to be addressed. Until now, most SARs exhibited simple reactive behaviors, were teleoperated, or were limited to conducting a single task in a uniform way. In this dissertation, a robot learner is proposed that accumulates instances of task demonstrations to generalize task dynamics during social interaction with users. As stated earlier in the section, instance-based learning provides an excellent framework for modeling tasks with very little prior knowledge. Also, learning different tasks is just an extent of maintaining multiple task-knowledge bases.

It was also observed that in clinical sessions many tasks are conducted that mimic children's play. Interactive play during childhood is crucial for one's cognitive development. Through playing, people learn basic interaction skills, collaboration, patience and build understanding of others [94]. With respect to playing with others, a shared

interest arises between playmates to make the play continuously entertaining, thus engaging the mind and creating opportunities for extended play over longer durations [45]. Thus, applications that we selected for evaluations with our robot learner include block stacking and inserting, sorting, and game applications (apps) on the tablet.

To further reduce perceptual uncertainties and increase robustness, we present methods that implement tasks on a touchscreen tablet that the person and the robot can share as a workspace. This approach is similar to research efforts that utilize simulated environments to acquire demonstrations, but a tablet provides a more intuitive and interactive workspace and has better accessibility. Tablets such as the iPad easily attract attention, provide convenient access to daily computing tasks, and are supported by a huge collection of mobile applications. Tablets are commonly available and their intuitive touch-based interface has replaced many traditional entertainment and educational products, such as televisions, video-game consoles, and textbooks. Tablets in classrooms have proven their power to better motivate students and increase students' learning performance [58, 80]. Articles also report how tablet computers are used to help children with disabilities and learning issues by actively engaging them with the device's attractive touchscreen interface and design [39, 109]. The secondary benefit of using tablets with robot learners is the ability to foster social interaction through child-robot play interaction that is directed by the child, thus moving towards interventions that can be translated outside of the clinical setting [43].

1.2 Objectives

1.2.1 Thesis Statement

A robot learner that utilizes an IIBL approach improves the performance and efficiency of general task learning, and when combined with the state-of-the-art mobile technology that provides personalized context for interaction, enhances the user's

experience for prolonged engagement of the task.

1.2.2 Research Hypothesis

The following presents overarching research objectives along with the hypotheses tested.

1) Ensure the IIBL framework effectively models tasks by utilizing demonstrations from the user even when an explicit model of the problem domain is difficult to elicit and is not amenable to complete mathematical modeling.

   - **Hypothesis I:** *IIBL provides comparable task performance against the average performance of the demonstrator.*

     Given a task T demonstrated by a teacher, the following are evaluated to test the hypothesis as outlined in [89]:

     – Evaluate the performance by measuring how well an agent A performs T.

     – Evaluate the behavior by comparing the actions executed by A while performing T against the teacher's actions when executing the same task T.

     – Evaluate the model by comparing the task model generated by A against the model the teacher used to provide demonstrations.

   - **Hypothesis II:** *The IIBL methods of modeling a retrieval function with adaptive weights reduces the workload, i.e., reduces the number of demonstrations required to achieve the same amount of system performance, compared to k-nearest neighbor (k-NN) which assigns equal weights to all features.*

2) Validate that the robot's learning behavior and performance impact the teacher's behavior and experience.

   - **Hypothesis III:** *The teacher's teaching strategy and behavior adapts to the robot learner's learning progress and performance.*
   - **Hypothesis IV:** *A session with robot learners has positive impact on the length and the number of interactions initiated by the user compared to a session with a person.*

1.3 Contributions

The key contributions of this work are summarized as follows:

- Developed an interactive instance-based learning approach for general task modeling for tasks that are given very little prior knowledge and are not always amenable to complete mathematical modeling.
- Developed an approach to keyword-based instance encoding and acquisition, a framework for accepting task knowledge from users.
- Applied multivariate linear regression and sensitivity analyses with neural networks as feature-weighting methods for robot learners and conducted human-robot interaction (HRI) studies that review user perception and adaptation.
- Designed a novel approach to using a touchscreen tablet as a shared medium for providing personalized contexts during human-robot interaction, and exploited the potential of a robot learner and tablet setting as educational, entertainment, and therapeutic platform.
- Conducted user studies during human-robot learning scenarios, and validated the utility of IIBL methods while analyzing the emergence of social behaviors

and the shift in teaching behaviors of the users.

1.4 Outline

This dissertation is organized as follows:

- Chapter 2 reviews prior work related to the dissertation research.
- Chapter 3 discusses our approach to modeling high-level task behavior through identifying and sequencing motion primitives. The limitations of applying the approach to robot learning are addressed, and an assistive application more suitable of using the algorithm is introduce.
- Chapter 4 presents the complete IIBL approach including identifying the issues of instance-based methods, addressing the issues through interactive learning, and designing feature-weighting methods for instance retrieval using multivariate linear regression and neural networks sensitivity analysis.
- Chapter 5 presents evaluations of the utility of IIBL methods as the robot learner engages in learning block-play tasks in a shared physical workspace.
- Chapter 6 presents evaluations of the utility of IIBL methods as the robot learner engages in learning tasks in a shared tablet workspace.
- Chapter 7 concludes the dissertation by summarizing the contributions and suggestions for future work.

CHAPTER II

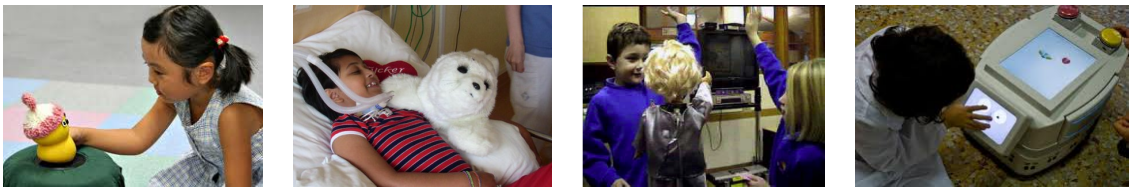
RELATED WORK

This chapter provides surveys of related research fields. The chapter is organized into four subsections: socially assistive robots, instance-based learning, interactive learning, and shared workspace. By designing a robot learner that utilizes an interactive instance-based learning (IIBL) and combining it with a shared workspace as a source for providing context, we attempt to develop a socially assistive robot learner that provides personalized experience to the users.

2.1 Social Assistive Robots

In recent years, various types of robotic tools have been developed and evaluated for therapeutic and rehabilitation purposes. The term *assistive robotics* generally referred to robotic devices that are built to assist in physical rehabilitation or support individuals with disabilities. Such examples include assistive manipulation for people with physical challenges [47, 69] or rehabilitation devices to aid in stroke recovery [24, 67]. *Socially-assistive robotics* (SAR), however, is a new concept that has not been clearly defined until recently. Feil-Seifer and Mataric [38] defined the term as robots that provide assistance through social interactions. Keepon and Paro [82, 121] are one of the first examples of SARs that created emotional and cognitive bonds simply by responding to the user's touch and sound. The roles of robots in pediatric therapy has been investigated by many research efforts [57]. Brooks and Howard [22] used a humanoid robot as a mediator in engaging and assessing people in a physical-therapy session. Iromec [77] was designed to engage children in various social and cooperative play. The authors carefully investigated the different types of disabilities and found the requirements necessary to engage each child in an interaction protocol.

The system was implemented by combining an autonomous mode and user-controlled behaviors, and its modular and configurable features provided flexibility in designing different play tasks. Figure 2 depicts examples of these SARs.



(a) Keepon (b) Paro (c) Robota (d) Iromec

Figure 2: People interacting with socially-assistive robots (SAR).

Of special interest is the use of SARs to engage children with autism spectrum disorder (ASD). In a case study with Robota [15], it was shown that therapy sessions with autistic children using robots, i.e., non-human entities, can be very effective in gaining the subject's attention for an elongated time. Scassellati [105] also mentions how robots generate a high degree of motivation and engagement in children with ASD, including those who are unwilling to interact with human therapists. His subjects showed positive social behaviors, such as touching, vocalizing, and smiling at the robot, which were rarely seen in natural day-to-day activity. Furthering this idea, robotic systems that possess the ability to imitate during play activity can promote a subject's social skills and mediate the transfer of learned knowledge to interaction with other people.

The Aurora/Iromec project is an ongoing study that investigates the possibility of using robotics as therapeutic educational-tools to engage autistic children in social interactions. Another robot used in this domain is Kaspar, which has been used in various study settings with autistic children [99]. The most interesting demonstration in this work is an imitation play, where the child and the therapist take turns mimicking Kaspar's expressions (Figure 3). Three children who typically refused to interact with other people, participated in the trial. The first child, after observing the robot

playing a tambourine, tried to imitate the action (Figure 3(a)). Moreover, after some time with the robot, the child reached out for the experimenter's hand, which could be interpreted as a first step towards interacting with another human. The second child incrementally transferred his interest from Kaspar to his therapist by comparing the robot's face with his teacher's, and he wanted to share his excitement (Figure 3(b)). The last subject had trouble tolerating other children in any play, but with Kaspar as a mediator, he gradually allowed another child to play turn-taking games with him (Figure 3(c)).

(a) (b) (c)

Figure 3: Kaspar acting as a social mediator allowing children to transfer learned social skills to other people [99].

Successful results from these studies motivate the research in this dissertation that focuses on the potential of interactive robotic systems. However, regarding the fact that nothing could be more crucial than user adaptation in these robots that provide therapies and educations for individuals with different levels of abilities and developmental stages, surprisingly low amount of attention was given in adapting these robots to the user's unique needs. Tapus et al. [115] developed a robot that would use either introverted and extroverted vocabularies to better motivate participants in physical rehabilitation exercises. For SARs to provide individuals with consistent and repetitive exposure to interactive activities, we believe that these robots need the ability to learn to engage in new tasks, combined with a shared workspace as a source for providing limitless personalized context. This research aims to develop a robot

leaner equipped with a framework that guides the acquisition of task experiences during play, provides generalization of the task, and generates robot's task behaviors by reusing prior experiences. This approach for learning was achieved through combining the methods of instance-based learning and interactive learning, which are reviewed in the following sections.

2.2 Instance-based Learning

Instance-based learning utilizes an analogical reasoning process that uses previous knowledge stored in memory to solve new problems. Case-based reasoning is a popular instance-based learning method that provides predictions to new problems from a baseline that similar problems have similar solutions [64]. By retrieving and reusing past solutions, the system can avoid the time necessary to derive solutions from scratch. This supervised learning method's generalization is provided by a distance function that measures similarity between instances. The k-nearest neighbors (k-NN), another popular form of instance-based learning, predicts a query's label using the k number of nearest instances in memory [3] and is often implemented within the CBR structure for instance retrieval and adaptation.

Instance-based learning methods store training instances in their raw form and postpone generalization until the query time. When a new instance is introduced, its classification relies on the stored data and its similarity with respect to previous instances. Machine learning with rule-based methods perform explicit generalization during training, and the instances used in the process are discarded afterwards. An approximation of a new instance is already formed during training, which hinders the explanatory behavior of the system. In instance-based methods, the generated prediction can be traced back to the original instances that provided the generalization. The instance-based methods' characteristics are summarized in the following [1]:

- Data driven: CBR solves new requests by combining information from stored

cases instead of computing rules that are difficult to extract. CBR allows the reasoner to propose solutions to problems quickly, avoiding the time necessary to derive those answers from scratch.

- Knowledge representation: CBR provides higher versatility in representing knowledge in more complex, symbolic forms [12]. Depending on the attributes to be represented, a number of instance formalism could be selected. In most CBR systems, the cases are usually represented as an attribute-value vector divided into two parts: problem and solution.

- Incremental learning: A CBR system is operational with only a small set of stored cases. New cases are collected during the time the system is used, increasing the system's problem-solving ability [2].

- Knowledge reusability: Since the cases are maintained in their original forms, they can be reused for solving multiple tasks by utilizing different similarity measures and adaptation methods. This is in contrast to eager methods in which generalization is already complete and cases used for training are discarded by the time a new instance query is made.

- Suitability for solutions spaces that are complex and incomplete: CBR systems can be applied to an incomplete model of the problem domain. The implementation not only identifies relevant instances but to fill in a partial case base with proper instances. Many domains are complex and not fully formalized, especially those that involve unpredictable human behavior. Even in these domains, the system needs to generate predictions, and with instance-based methods, the system can make assumptions and predictions based on what worked in the past without having a complete understanding of the domain.

Many prior works used instance-based learning for robot learning, planning and

action-selection problems [40, 101, 120]. In [88] and [32], the authors collected demonstrations from the users through a simulated environment that were used toward training the case-based reasoning system. In most of these systems, it wasn't their focus to develop a general learning system, and therefore, the processes of representing, acquiring, and retrieving instances required domain knowledge that were supplied by the experts. When the instance-acquisition process was automated, the process was often separated from an actual system deployment, limiting the capability of the system to update or renew already existing instances. This dissertation have addressed these challenges through interactive methods of machine learning. Combined with interactive machine learning, the explanatory behavior of instance-based methods facilitates learning by notifying the teacher about the exact reasoning process of the learner. The instance-based learning steps within the overall system framework are depicted in Figure 4. This flowchart highlights the four steps of case-based reasoning: case retrieve, reuse, revise, and retain.

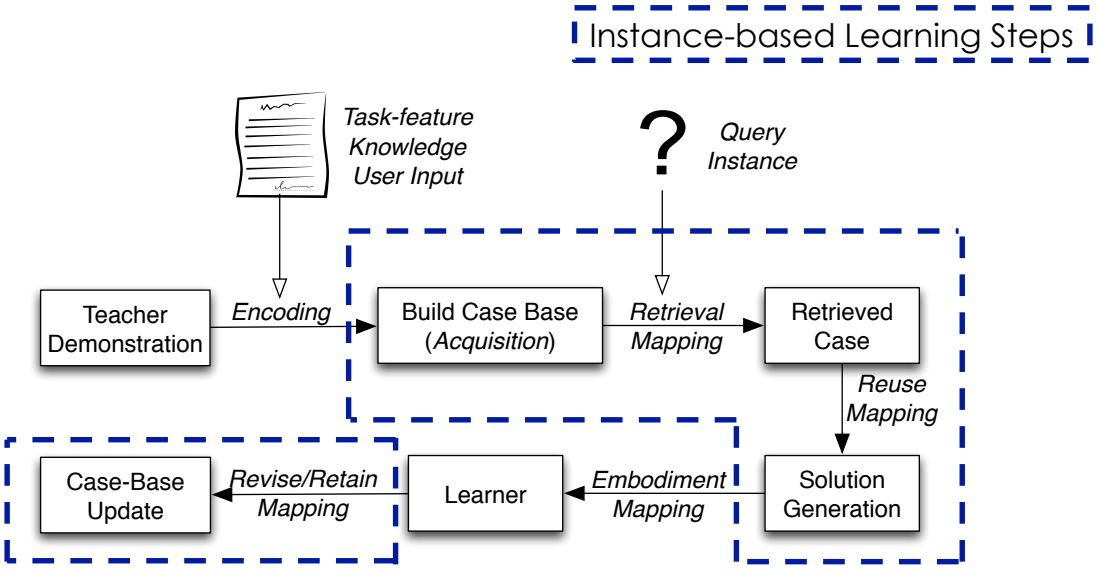


Figure 4: Instance-based learning acts as an overarching framework for general task learning.

The main limitation of instance-based learning, that the generalization can only be made to an extent of a task space covered by its instances, can be improved through an interactive system that monitors the system's state in real time and provides necessary inputs to better cover the given problem space. In the next section, a review of interactive machine learning is presented.

2.3 Interactive Machine Learning and Learning from Demonstration

Interactive machine learning (IML) places humans in the process of designing, training, and evaluating machine-learning systems [37, 123]. In IML, humans provide inputs to the system and monitor the output that in turn influences what inputs they will provide next. The action of providing human input is often referred to as teaching, and the "teaching" occurs during a system deployment which makes the learning process interactive. Therefore, learning from demonstration (LfD) with social robots naturally foster an IML setting. LfD is a method for programming new skills to the robot by providing human demonstrations [4, 6]. Research efforts in agents learning from human teachers are not only limited to teachers providing inputs. Robots can also ask for help after a failure [86] or actively request to resolve ambiguity [26]. A transparency mechanism can allow the teacher to better monitor the robot's state and progress [30, 116, 117].

For instance-based approaches, IML and LfD can enhance and assist the process of acquiring and maintaining knowledge. In most knowledge-intensive systems, CBR is applied as a knowledge retrieval and management tool in the form of KI-CBR (knowledge-intensive CBR) where the case base is preloaded often using ontology [97]. In [100], KI-CBR is used to generate dialogue responses during human-encounter episodes in a hotel. The hotel's ontology is preloaded with information categories, such as hotel facilities, tourist places, and public transportations. Recently, there have been successful efforts in applying LfD techniques to automate the process of

case acquisition in CBR, sometimes referred to as a *lazy-LfD* approach. In [87], the authors solved the issue of populating a case base with plans through LfD for generating planners for real-time strategic games. In [62], a home-service robot generated action sequences for fetching objects based on past cases after receiving a request from a user. Here, the system actively manages the case library by modifying and improving retrieved cases with the help from the user. A data-driven CBR was developed in [32] through crowd-sourcing. In this work, the proposed system collected 82,479 cases during human-human collaborative task in a virtual reality environment. Afterwards, the case base was used towards a similar task conducted in the physical world with a human-robot team to generate robot behavior. The case base was occupied autonomously by extracting the features from demonstrations.

This dissertation proposes an interactive instance-based learning (IIBL) framework that utilizes IML to actively engage the user in the process of knowledge encoding and acquisition during human-robot interaction on a shared workspace (Figure 5). By providing a visualization of the system's current state and performance, such as through a robot's behavior, programming a machine-learning system becomes more accessible to the end-users. Compared to previous works using LfD for providing batch instances prior to an actual system deployment, IIBL collects and maintains data while interacting with the teacher. Unlike other ML methods that turn into a black-box after training, the instance-based approach's explanatory behavior provides transparency to the system by informing the user with exact instances that generated the predictions. We have observed through user studies that the teacher's behavior adapts to the learner's progress and performance.

2.4 Shared Workspace

There are several issues that hinders SARs in providing long-lasting engagement. Some of these issues come from the fact that many of these robots are tele-operated,

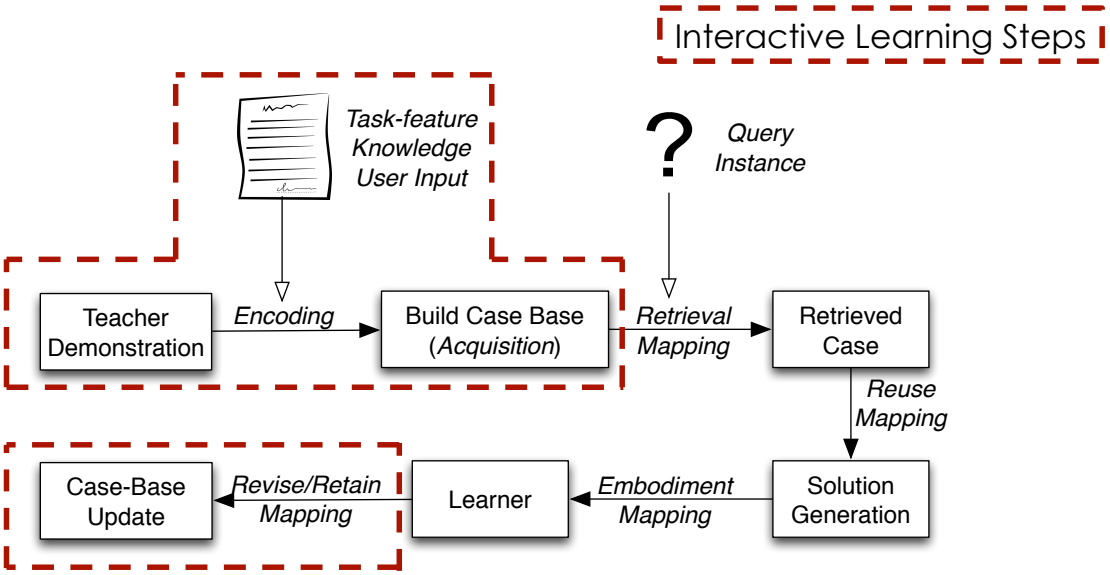


Figure 5: In IIBL framework, the user actively engages in the process of knowledge encoding and acquisition during human-robot interaction.

or exhibit very limited reactive functions, such as reacting to touch or sound, and most importantly, these robots have very limited source of providing personalized context. The context between the robot and the child, or any individual, should be something that can be renewed freely during an interaction and throughout the life time of an individual. And for that matter, we believe that robots should be able to learn to engage in these different tasks that arise during an interaction.

The scope of this research is in applying a robotic agent to learn and conduct tasks with humans in a shared workspace. In this dissertation, a shared workspace is defined as a proximate area between the person and the robot where the task events take place. For instance, when the task is block stacking, the shared workspace is the area in which the blocks are populated and manipulated. A physical presence of all entities is assumed in a shared workspace. Many works report the effectiveness of deploying physically embodied agents versus simulated or non-embodied artificial intelligence in facilitating the emergence of social behaviors and in encouraging engagement of

the users [61, 72, 122].

In this dissertation, the shared-workspace concept extends beyond the physical space and encompasses virtual spaces defined on a physical device. Engaging in tasks defined on a shared touchscreen tablet is one such examples in which the person and the robot take turns manipulating objects on the screen. It is reported that robots can enhance user experience through functioning in conjunction with applications (apps) on smart devices. Popchilla [14] (Figure 6(a)) combines an interactive drawing application with a robot that generates motion and sound responses to user's input on the tablet. The robotic music listening companion [55] (Figure 6(b)) produces on-beat motions to music played from a smart phone. The robot's rhythmic behavior makes the person feel like they are sharing the experience, and the person perceives the event as more enjoyable. Other examples include robots to which users can mount their smart phones, which engages the users by animating their devices [13] (Figure 6(c)). The tablet can also be mounted on a mobile platform for telepresence robots (Figure 6(d)) [68].

As a research platform, touchscreen devices function as shared workspaces and reduce perceptual uncertainties. In LfD, there are several methods for recording teacher's demonstration [4, 20, 85]. The uncertainty of environmental perception always poses a difficult challenge, including tracking teacher's motions or recognizing a human's social cues. The tablet platform reduces such uncertainty since the touchscreen provides quantified sensor data from the gestural behavior of the user. The development environment and already available apps on the market facilitate the process of designing and implementing a task with controllable modalities. Such benefits of deploying a touchscreen-based medium for studying interactions between human and robot are discussed in [10]. In their work, the robot is programmed to replay pre-assigned motions for a simple task, and the research is focused on emerging social behaviors of the user.

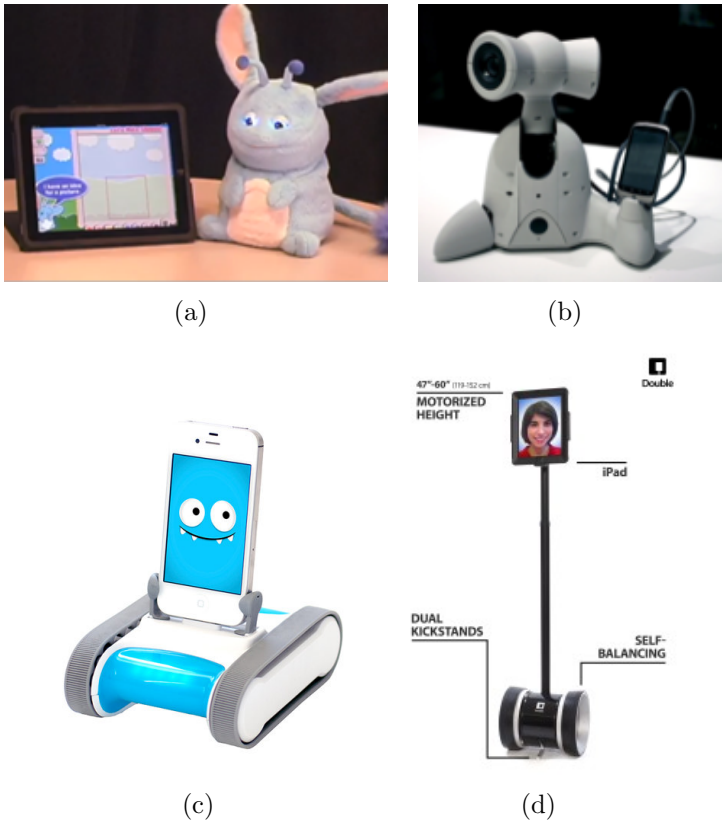


Figure 6: Examples of robot platforms that enhance user's experience with smart devices. (a) Popchilla [14], (b) Robotic music listening companion [55], (c) Romo: smartphone mobile robot [13], (d) Mobile robotic telepresence platform controlled by a tablet.

However, the above systems have limitations in providing continuous engagement because they are only applicable to proprietary mobile apps and their functions are static and reactive. This dissertation research addresses the efficacy of coupling commercially available mobile devices as a shared workspace with a robot learner to provide various contexts for human-robot interaction studies [93] and to effectively bring a personalized experience that matches each individual's needs and preferences.

CHAPTER III

MODELING TASK BEHAVIOR AS SEQUENCES OF PRIMITIVES

Conducting tasks together on a shared workspace require manipulative behaviors. From an observational study of children's object-play activities, we have identified motion primitives that were repetitively observed across different play scenes. A sequence of these motion primitives provides an understanding of object-manipulation behaviors that form the basis of task learning. In this chapter, we first report the findings from the observational study and discuss the approach to recognizing and sequencing motion primitives. The approach is then advanced towards online training of gesture primitives for an assistive device that takes training inputs from individual users with limited upper-body mobility. The training inputs are used to calibrate the device to address each user's motion range in real time. At the end, limitations of the approach presented in this chapter are discussed, and a new method that addresses these limitations is proposed.

3.1 Introduction

Most research that focuses on learning manipulation tasks attempts to address the problem associated with general learning. Modeling such tasks is often done by means of measuring devices, such as motion capture suits, feedback actuators, or haptic devices. In addition, many contributions in gesture recognition also use colored gloves [60] or data gloves as discussed in [71]. In contrast, the intent of the proposed approach is to apply dynamic pattern-recognition methods using only visual information without further aids. Though vision sensors are noisy and sensitive to various changes in

the environment, camera modules are low-cost devices that are usually readily available in most robotic platforms. To improve drawbacks associated with vision, proper preprocessing methods should be applied to image frames.

Although there are a number of gesture-recognition algorithms in existence, one of the most popular algorithms has been applied to recognizing American sign language [114]. The system used a single camera to track hands in real time and adapted a discrete hidden Markov model (HMM) to understand a full-sentence construction. The authors successfully demonstrated the feasibility of recognizing a series of complicated gestures with their proposed system. HMM has also been used in recognizing image sequences of six different tennis strokes among three subjects [130]. Darrell and Pentland [36] used the dynamic-time-warping algorithm to match the interpolated responses of several image templates. Most recently, a low-cost vision system providing depth information (KINECT) has gained popularity in gesture-recognition society [59]. KINECT can track the human body and limbs in real time and recognize certain motions that are trained using joint-state data.

Earlier results of our research have shown that through observation, a robot can successfully sequence low-level motion segments to form a complete play action [56, 119]. These efforts, however, were more focused on identifying the functional and relational characteristics of play and developing a method to transfer the relative information to the robot. In recent work [92], an in-depth study was conducted on human's play motions and a method for recognizing and sequencing motion primitives was presented. As seen in many related research efforts, HMM-based methods provide reliability and simplicity in recognizing sequential patterns, and thus the algorithm has been selected for modeling play-task behaviors in this dissertation. In the following sections, the result of the object-play-primitive research is presented, and a detailed description of the motion-recognition algorithm is provided.

3.2 Object-play Primitives Research

Baranek et al. [8] describe in their work a subset of toy manipulations that could be used towards screening a child's developmental stage. The list contains the behaviors shown in the first five levels of object play: *grasp, rub, shake, bang, mouth, roll, pull apart, stack, scoop, pour, cover,* and *join.* Based on this list, a study was conducted with public sources from the web to identify the kinds of basic motions that form these manipulations when people interact with various types of toys. With regards to a robotic playmate, these basic motions are further defined as *play primitives.*

Correlational studies often involve longitudinal and cross-sectional research. For instance, Baranek et al. [8] used a retrospective method to analyze home movies of 32 infants at nine to 12 months of age. The researchers also compared the infant's object-play behaviors within the three groups: infants with autism, infants with other developmental-delay conditions, and typically developing infants. On the other hand, the intent of this proposed research is to analyze the primitive components that are shared in object play throughout the subject's developmental conditions and stages. Therefore, the gathered video sequences consist of various toy-manipulating scenes irrespective of the subject's age or current developmental issues. As a start, 25 video sources were collected from YouTube. Various play scenes were considered including building blocks, stacking cups, inserting blocks, and hammering tables (Figure 7). The videos consisted of 32,676 valid frames where only the frames that contained an image of object manipulation were classified as valid. The identified play primitive in each valid frame was then used to calculate the ratio of primitives observed during the session.

The observations made during this study are depicted in Figure 8. As can be seen, many of the play scenarios begin with picking up an object in an upward direction. The seven most distinct primitives (94.21%) found from the study are the renditions of the behaviors in the list of Baranek *et al.* (Figure 8(a)). For instance, the *stacking*

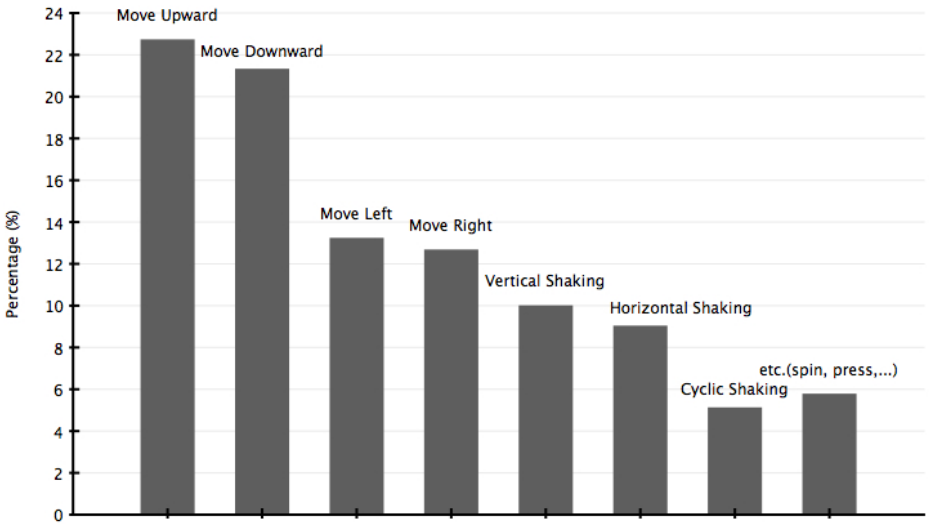
Figure 7: Video study reveals most frequently observed play primitives.

behavior from the list is a sequence of moving up, left/right, and down. Similarly, *banging* behavior could be modeled as a sequence of repetitive vertical shaking. Other primitives that were observed less frequently are spinning, pressing, and hugging. In addition to these primitives, the manipulated objects were observed in three types of final resting states (Figure 8(b)): insert (44%), stack (32%), and drop (24%). A disappearing of a play object characterizes the inserting primitive. The stacking and dropping primitives are distinguished by whether the object is placed on top of another object or not.

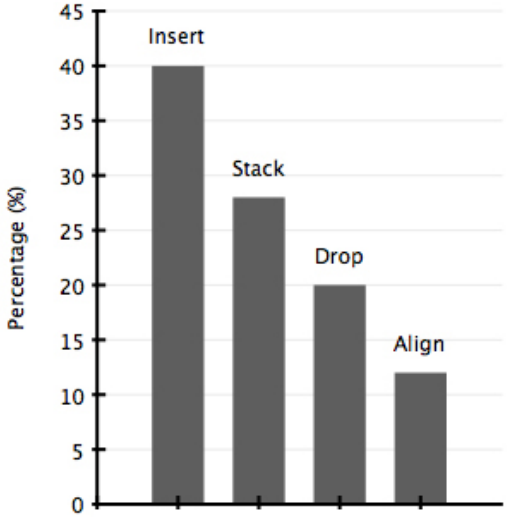
The play primitives that are implemented throughout this research are based on the statistics learned from the above study. The final play primitives are shown in Figure 9.

3.3 Object-play Behavior Modeling

Hidden Markov modeling (HMM) methods provide a probabilistic framework for modeling a time series of multivariate observations. The power of the algorithm comes



(a) Percentage of each play primitive observed during an object manipulation.



(b) Percentage of the final state of the manipulated object observed in the video samples.

Figure 8: Statistics of the observed play primitives from the video study of 25 play scenes.

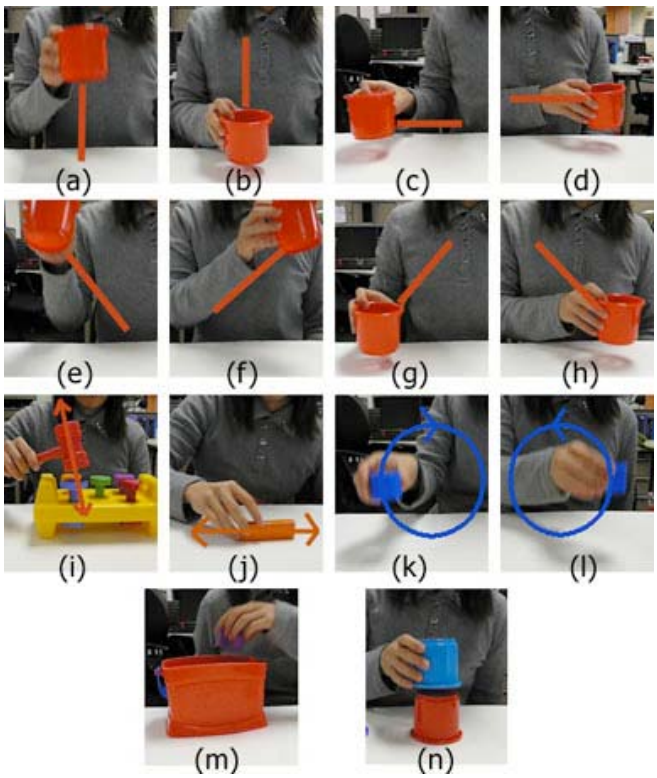


Figure 9: The most frequently observed basic motions are defined as play primitives. (a)-(b) vertical movement primitives, (c)-(d) horizontal movement primitives, (e)-(h) diagonal movement primitives, (i)-(l) repetitive vertical/horizontal/circular movement primitives, (m) insert, and (n) stack primitives.

from the characteristic that defines a Markov process. In a Markov process, the conditional probability distribution of the future event only depends on the current event and not on the sequence of predecessor events. In modeling play actions, it might not be valid to assume that the entire play sequence has a Markov property. However, the assumption is still useful when considering the constant changes in motion gradients and object appearances that form the play motions.

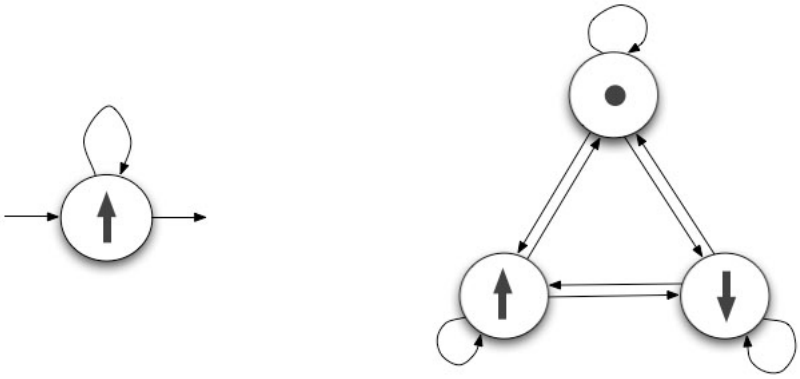
3.3.1 Hidden Markov Models

Two primitive examples are shown in Figure 10. The first one is a simple upward motion, and the other is a repeated up- and down-hammering motion (Figure 10(a)). If these actions were to be modeled by tracking the object's center location in every

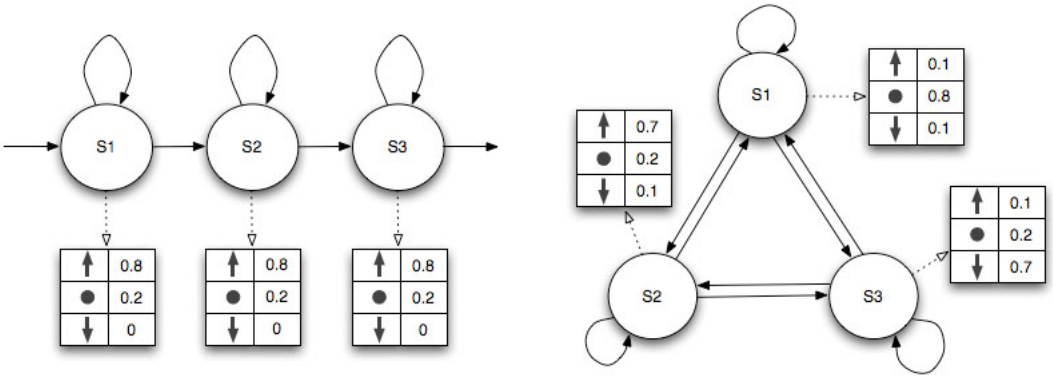
frame, the motion gradient of the upward motion would be a sequence of ↑↑↑↑, and the hammering motion would be a repeated sequence of ↑↑ • ↓↓ • in which the • means a steady position of the object. The ↑, ↓, and • correspond to the states in Figure 10(b). However, a gesture cannot be executed exactly the same every time. The length of each state and the angles of the motion gradients will vary. In addition, tracking the objects via vision sensors introduces noise that affects the system performance in retrieving motion gradients from each frame. Intuitively, upward motion should only consist of ↑↑↑ ... as stated earlier, but results like ↑↑↖↑ • ↑ ... are often observed in experiments. This ambiguity in gradient extraction becomes more obvious when modeling primitives like Figure 9(i)∼(n). These primitives involve multi-directional gradients that are constantly changing through time with no fixed length. Thus, the Markov models in Figure 10(c) was introduced. In these models, each state emits possible outputs in the form of discrete probability distribution. These outputs are called *observations*. Since the exact order of the states cannot be retrieved in which the observations are emitted, such representation is called hidden Markov modeling.

Extending the example shown in Figure 10, the HMM algorithm for recognizing play primitives and the process of training the HMM models are discussed next. According to Rabiner [96], the HMM poses three questions: evaluation, decoding, and learning. In the following paragraphs, the evaluation and learning methods that are used to recognize and model play primitives are explained in detail. The following notations are used throughout the remainder of this section:

(a) Upward and vertical-shaking play primitives



(b) Markov chains of the two primitive examples with state transition.



(c) Hidden Markov models of the above two examples. The probabilities of generating observational symbols from each state is shown in the table.

Figure 10: Two play primitive examples are shown with corresponding Markov chains and hidden Markov models.

N: The number of states in a hidden Markov model.

T: The observation-sequence length.

S: The state sequence, $S = \{s_1, s_2, s_3, ..., s_T\}$, where s_t is the state at time t.

a_{ij}: The state-transition probability from state s_i to s_j.

$b_j(k)$: The probability of generating an observational-symbol k from state s_j.

A: The state-transition probability distribution, $A = \{a_{ij}\}$.

B: The observational-symbol probability distribution, $B = \{b_j(k)\}$.

π: The initial-state probability distribution.

λ: The notation for a parameter set of a hidden Markov model, $\lambda = (A, B, \pi)$.

The evaluation problem is stated as follows: Given a model λ and an observation sequence O in a length of T, what is the probability that the model actually generated those observations? Among multiple hidden Markov Models $\lambda_1, \lambda_2, ..., \lambda_n$, which model best describes the given observation? The answer to these questions boils down to one: Which model has the better probability $P(O|\lambda)$?

The first step in calculating $P(O|\lambda)$ is to evaluate all possible hidden state sequences and calculate the probability of the model generating the observed sequence:

$$\begin{aligned} P(O|\lambda) &= \sum_{all\, S} P(O|S,\lambda)P(S|\lambda) \\ &= \sum_{all\, S} \prod_{t=1}^{T} a_{s_{t-1}s_t} b_{s_t}(O_t), \qquad \text{where} \quad a_{s_0 s_1} = \pi_{s_1}\,. \end{aligned} \tag{1}$$

The term inside the large product is the probability of transitioning to the state s_t from $s_{(t-1)}$ at time t multiplied with the probability of observation O_t being emitted from the state s_t. The product of these terms over the whole length of time T gives the probability of the state sequence S and the observation sequence O occuring together. Now, summing this probability over all possible state sequences gives the final

resulting $P(O|\lambda)$. Since there are N^T combinations of state sequences, $(2T-1)$ number of multiplications for each state sequence, and (N^T-1) number of additions, the above equation requires the order of $2^T N^T$ calculations that increases exponentially over time.

A more effective way to calculate $P(O|\lambda)$ is to use the forward-backward algorithm [96]. The algorithm efficiently computes the values that are required to obtain the posterior marginal distributions first by traveling forward in time and then going backwards in time. Here, forward and backward variables are introduced:

α: The forward variable α is the probability of the partial observation sequence $O_p = O_1 O_2 O_3 ... O_t$ until time t and state s_i at time t given the model λ, i.e.,

$$\alpha_t(i) = P(O_p, s_t = s_i|\lambda)\,. \tag{2}$$

β: The backward variable β is the probability of the partial observation sequence $O_p = O_{t+1} O_{t+2} O_{t+3} ... O_T$ from time $t+1$ to T and state s_i at time t given the model λ., i.e.,

$$\beta_t(i) = P(O_p, s_t = s_i|\lambda)\,. \tag{3}$$

The forward variable is first initialized as a probability of emitting the first observation symbol O_1 in state s_i:

$$\alpha_1(i) = \pi_i b_i(O_1), \qquad 1 \leq i \leq N\,. \tag{4}$$

The probability of getting to a state through all paths is calculated by computing the probability of each path to the state and summing all the probabilities (the induction step):

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i) a_{ij}\right] b_j(O_{t+1}), \qquad 1 \leq t \leq T-1, \quad 1 \leq j \leq N\,. \tag{5}$$

The probability of the sequence given the HMM λ is then the sum of the partial probabilities at time T:

$$P(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i)\,. \tag{6}$$

The backward algorithm, similar to the forward algorithm is as follows:

$$\beta_T(i) = 1, \qquad 1 \leq i \leq N\,, \tag{7}$$

$$\beta_t(j) = \sum_{i=1}^{N} a_{ij} b_i(O_{t+1}) \beta_{t+1}(i), \qquad t = T-1, T-2, ..., 1, \quad 1 \leq j \leq N\,. \tag{8}$$

Next, the values for each model $\lambda = (A, B, \pi)$ need to be determined. The learning algorithm generates a solution λ by adjusting the model parameters A, B, and π to maximize the probability $P(O|\lambda)$ given the observation O. No analytical method is known to solve this problem, but by using the Baum-Welch algorithm [96], the parameters could be refined iteratively. To start, a new variable γ is introduced:

γ: $\gamma_t(i,j) = P(s_t = s_i, s_{(t+1)} = s_j|O, \lambda)$ is a posterior probability of being in state s_i at time t and s_j at time $t+1$ given the model λ and the observation sequence O. $\gamma_t(i) = P(s_t = s_i|O, \lambda)$ is then defined as the probability of being in state s_i at time t given the observation and the model. Thus, $\gamma_t(i) = \sum_{j=1}^{N} \gamma_t(i,j)$.

For a given sequence O, the probability of transitioning from state s_i to s_j is

$$\begin{aligned}\gamma_t(i,j) &= P(s_t = s_i, s_{(t+1)} = s_j | O, \lambda) \\ &= \frac{P(s_t = s_i, s_{(t+1)} = s_j, O|\lambda)}{P(O|\lambda)} \\ &= \frac{\alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)}{\sum_i \sum_j \alpha_t(i) a_{ij} b_j(O_{t+1} \beta_{t+1}(j)} \,.\end{aligned} \tag{9}$$

Using the above formula, parameters are reestimated until the difference between the two sequential-model estimations, $\log P(O|\lambda_{new}) - \log P(O|\lambda_{prev})$ falls below some threshold. The new estimations of a_{ij}, $b_j(k)$, and π are then obtained by the following formulae:

$$\bar{a}_{ij} = \frac{\sum_{t=1}^{T-1} \gamma_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)} \,, \tag{10}$$

$$\bar{b}_j(k) = \frac{\sum_{t \in O_t = k} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)} \,, \tag{11}$$

$$\bar{\pi}_i = \gamma_1(i) \,. \tag{12}$$

In this work, hidden Markov models for each play primitive in Figure 9 were trained and recognized. In the following sections, the necessary steps for training the HMM are explained. To extract features for the hidden Markov modeling, each image frame is processed to track objects in the scene. The feature vector includes information of the motion gradients and the size variation of the object. Later, the vector is quantized into a discrete symbol using the codebook. The converted symbols form an observation sequence that is loaded to the Baum-Welch algorithm to train the models.

3.3.2 Motion-gradient-feature Extraction

Choosing the right features that describe the play primitives leads to a successful model training and recognition in HMM. Lets say there is a method to track moving objects in each frame. The first thing to be considered is the object's position in every frame. However, the absolute pixel value of the position is meaningless. What defines the movement of an object is the directional difference in the object's center of mass in adjacent frames. The term *motion gradient* is used to describe the normalized direction of the motion. Besides the x and y directional changes of the object movement, the disappearance of an object is also a critical feature in determining whether the object has been inserted into another. The disappearance of an object is observed as a gradual decrease in the visible size of an object. Tracking and recognizing the objects in a scene is explained in the next section.

The motion gradient vector $(dx_i, dy_i)^T$ is computed using two adjacent frames. The center of mass $(m_{x_i}, m_{y_i})^T$ is defined as

$$m_{x_i} = \frac{\sum_{(x,y)\in D_i} x}{N_i}\,, \qquad m_{y_i} = \frac{\sum_{(x,y)\in D_i} y}{N_i}\,. \tag{13}$$

In the above equation, D_i represents the region of the detected object in the i-th frame, and N_i is the number of pixels in region D_i. The gradients $(dx_i, dy_i)^T$ are calculated using the following formulae:

$$\begin{aligned} dx_i &= \frac{(m_{x_i} - m_{x_{i-1}})}{\sqrt{(m_{x_i} - m_{x_{i-1}})^2 + (m_{y_i} - m_{y_{i-1}})^2}}\,, \\ dy_i &= \frac{(m_{y_i} - m_{y_{i-1}})}{\sqrt{(m_{x_i} - m_{x_{i-1}})^2 + (m_{y_i} - m_{y_{i-1}})^2}}\,. \end{aligned} \tag{14}$$

Note that the magnitude of the motion gradient is normalized to extract the directional vector only, i.e,

$$\sqrt{dx_i^2 + dy_i^2} = 1\,. \tag{15}$$

Next, the ratio of the object spread is calculated from the change in object size defined as

$$\begin{aligned} d\sigma_{x_i} &= \frac{\sum\limits_{(x,y)\in D_i} (x - m_{x_i})^2 \Big/ N_i}{\sum\limits_{(x,y)\in D_{i-1}} (x - m_{x_{i-1}})^2 \Big/ N_{i-1}}\,, \\ d\sigma_{y_i} &= \frac{\sum\limits_{(x,y)\in D_i} (y - m_{y_i})^2 \Big/ N_i}{\sum\limits_{(x,y)\in D_{i-1}} (y - m_{y_{i-1}})^2 \Big/ N_{i-1}}\,. \end{aligned} \tag{16}$$

Here, $(d\sigma_{x_i}, d\sigma_{y_i})^T$ is the ratio of the variance between the i-th and $(i-1)$-th frames. As stated above, the size variance is used to determine the play object's final resting state. According to the study discussed in Section 3.2, these three states, i.e., insert, stack, and drop, were one of the most popular objectives in turn-taking play tasks. Therefore, it is carefully assumed that the toy object, with high probability, is inserted, stacked, or dropped during a play session. The inserted state is trained as a gradual disappearance of the object after a downward action towards another object. The stacked and dropped states are recognized with the same hidden Markov model, but the two are distinguished afterwards by identifying whether the play object was placed on top of another object or not.

From the four features introduced above, a 4-dimensional motion vector

$$\overrightarrow{M_i} = [dx_i, dy_i, d\sigma_{x_i}, d\sigma_{y_i}]^T \tag{17}$$

is derived for every frame resulting in a vector sequence $\overrightarrow{M} = \overrightarrow{M_1}\overrightarrow{M_2}\overrightarrow{M_3}...\overrightarrow{M_T}$.

3.3.3 Preprocessing: Tracking Objects

Tracking object movements gives an idea of how the subject is manipulating the object. Since the toy objects used in this research have solid and high-saturated colors, a color-recognition algorithm is used as the method of tracking. The image frames are processed with a histogram-back-projection algorithm to provide stable recognition of the colored objects.

When play is initiated, the system first observes the entire scene. Since the main focus of this research is to understand how subjects interact with toy objects, the objects are tracked in the image scene versus the subject's body parts. The vision module focuses on detecting objects using color segmentation. Since many toy objects share similar, commonly occurring colors, the vision system is trained to classify pixels into these color classes using hue-saturation histogram models.

A back-projection is a way of representing how likely each pixel fits the distribution of pixels in a histogram model [17, 18]. The calibration step is used to cope with variations in illumination conditions, and the histogram models are redefined during this step. Once the histogram models are computed, the models are used to assign a probability value to each image pixel in subsequent video frames. For every new frame, the hue and saturation value for each pixel is determined, and each color histogram is used to assign a probability to the pixels. In this proposed toy hue-saturation histogram model, if C is the color of the pixel, and T is the probability that a pixel is a toy, then this probability map generates $p(C|T)$. $p(C|T)$ is the probability of drawing the color C if the pixel actually is a toy. Combining the total probability of encountering a toy-colored object in a scene $p(T)$ with the total probability of encountering the range of toy colors $p(C)$, $p(T|C)$ is computed using the following

Bayes' theorem:

$$p(T|C) = \frac{p(T)}{p(C)} p(C|T)\,. \tag{18}$$

This process allows the system to identify, with high probability, all toy objects within the scene. Among the multiple objects detected from this process, the first object upon which an action is taken is labeled as the *primary object*. This object is then tracked until it comes to a complete stop, and the motion-gradient data is recorded during the tracking. The output example is shown in Figure 11.

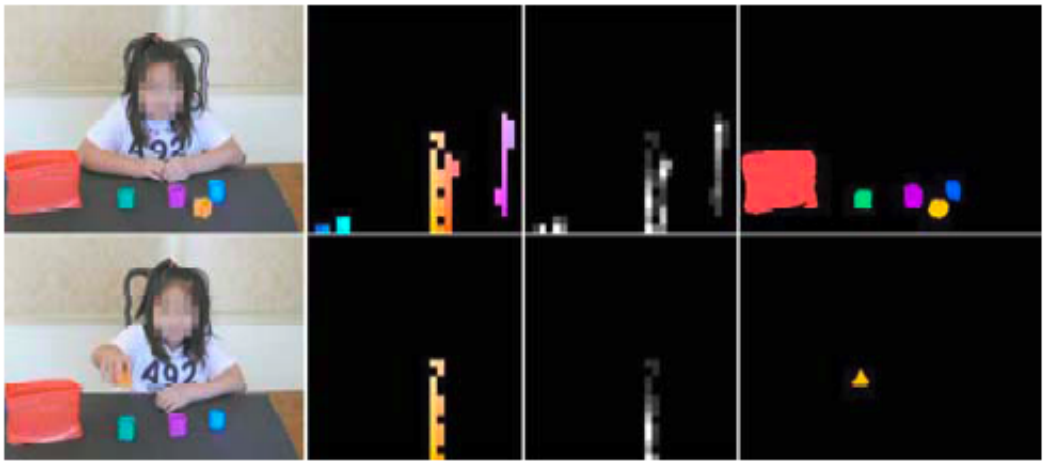


Figure 11: Preprocessing results of a play scene. (Top) From the left: The initial play scene, hue-saturation histogram color map, hue-saturation histogram probability map, and the detected objects. (Bottom) From the left: The primary object being tracked, hue-saturation histogram color map of the primary object, hue-saturation histogram probability map of the primary object, and the detected primary object.

3.3.4 Codebook and HMM Topology

Hidden Markov models have an advantage in modeling sequential patterns such as speech. The proposed work models a behavior as a sequence of object-play primitives. In effect, a play behavior is modeled with temporally sequenced play primitives,

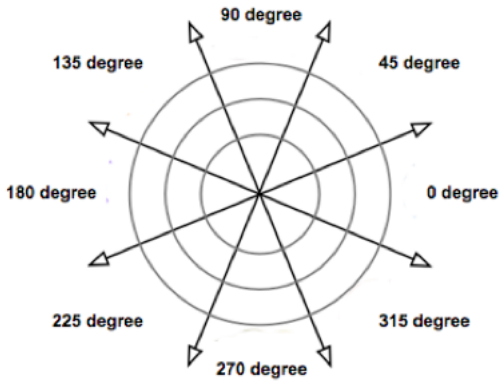
which is analogous to the representation of a word using a sequence of phonemes in speech recognition. It is, therefore, understandable why these techniques developed for speech recognition would perform well in modeling human gestures.

In this section, a method to convert autonomously extracted features from Section 3.3.2 into discrete symbols for a hidden Markov system is presented . First, the motion vector sequence $\overrightarrow{M_1}\overrightarrow{M_2}\overrightarrow{M_3}...$ is translated into discrete observation symbols $O_1, O_2, O_3...$. The reference for the translation, i.e, the codebook, is built with respect to the feature space that is divided into 18 clusters (Figure 12). The first classification process uses the directional information $(dx, dy)^T$ to calculate the gradient angle that falls into one of the quantized eight regions (Figure 12(a)). The size ratio $(d\sigma_x, d\sigma_y)^T$ that is used to distinguish the final-resting-state primitives forms another category. The final 18 codes are depicted in Figure 12(b).

The simple structure of this codebook enhances the extendability of the proposed recognition system. The motion gradient and the codebook are designed so that different motion primitives for various objects could be easily added to the system. The two types of HMM topologies that were introduced in the previous HMM example (Figure 10) are used to model the play primitives (Figure 13). A left-right model, also known as the Bakis HMM, is used to model non-cyclical primitives, such as the first eight primitives and the two final-state primitives (Figure 13(a)). The repetitive gestures such as the four shaking primitives are modeled using a cyclic-left-right HMM with three states (Figure 13(b)).

The recognition problem is now reduced to finding the HMM with the highest probability:

$$\lambda = \arg\max_{\lambda_l} \left[P(O|\lambda_l) \right] . \tag{19}$$



(a) Eight directional regions.

Symbol	Direction	Size ratio	Symbol	Direction	Size ratio
1	0°	> 0.83	10	0°	< 0.83
2	45°	> 0.83	11	45°	< 0.83
3	90°	> 0.83	12	90°	< 0.83
4	135°	> 0.83	13	135°	< 0.83
5	180°	> 0.83	14	180°	< 0.83
6	225°	> 0.83	15	225°	< 0.83
7	270°	> 0.83	16	270°	< 0.83
8	315°	> 0.83	17	315°	< 0.83
9	Steady		18	Out-of-sight	

(b) Classification of observation symbols.

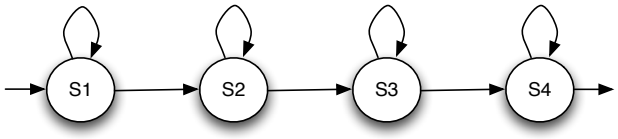
Figure 12: Codebook for translating motion vectors into observational symbols.

The result with a probability under a threshold is discarded as an unknown. The final primitives recognized with sufficient probability are sequenced to form a complete play action. The overall structure of the play-primitive-recognition system is depicted in Figure 14.

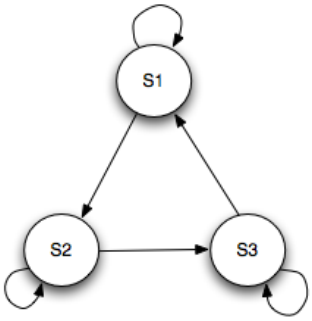
3.3.5 Evaluation and Discussion

3.3.5.1 Experimental Setup

For training the HMMs, 20 play scenarios were collected from three adult participants. For testing the system, 30 sessions were videotaped from three adults, and 70



(a) Left-right model used for modeling linear primitives.



(b) Cyclic-left-right model for modeling repetitive primitives.

Figure 13: Hidden Markov model topologies.

sessions were collected from three child participants. Three different cameras were used in three different environments to verify the system's capability for tracking play primitives in low- and high-resolution images and to measure the algorithm's adaptability to illumination changes (Figure 15). The play tasks consisted of inserting blocks into bins, stacking cups, hammering pegs, and dropping toys. Each session varied in length, some with simple motion trajectories and others with longer interaction time with the objects. The sessions lasted for two to 30 seconds.

Using the HMM structure stated in Section 3.3.4, five to 15 samples for each primitive were collected from the training dataset. The vector sequences received from the motion-gradient-extraction module were quantized into discrete observation symbols using the customized codebook. The sequence of observation symbols were then simultaneously used as inputs to the Baum-Welch algorithm that ran until the values of the HMM converged. The experiment result shows that the system took an average of four iterations to achieve over 95% convergence.

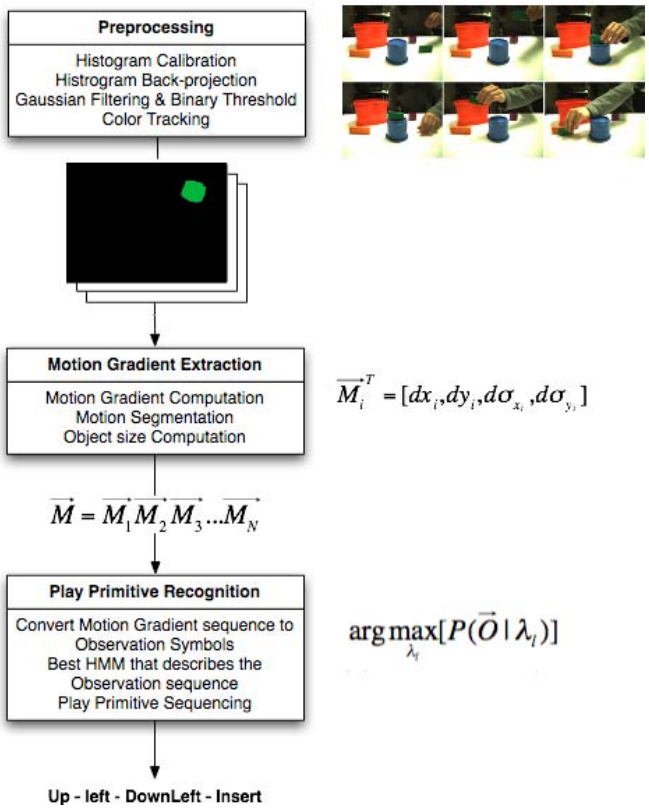
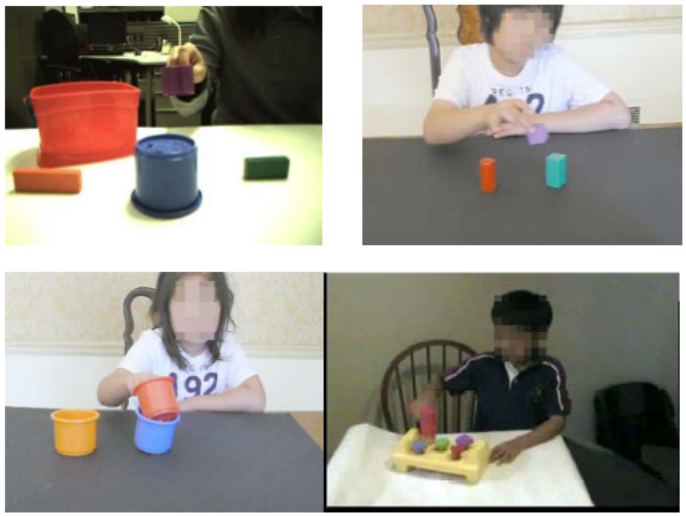


Figure 14: Training the hidden Markov models and recognizing the play primitives.

The 30 test scenarios were collected from three adults in the same environment as the training dataset. The three children each participated in 20 to 25 sessions, and the sessions were videotaped at each participant's home. The content of the play sessions ranged from a very simple pick-up and insert operation to a more complicated task shown in Figure 16. The test dataset was used to examine the performance of the system in recognizing the play primitives and correctly sequencing them together. The average frame rate of the overall algorithm was 21.6 frames per second (fps).

3.3.5.2 Results and Discussions

The recognition rate of each play primitive is shown in Figure 17, and a detailed result is illustrated as a confusion matrix in Figure 18. The overall recognition was performed with 86.88% accuracy. The average recognition rate was 94.51% for the



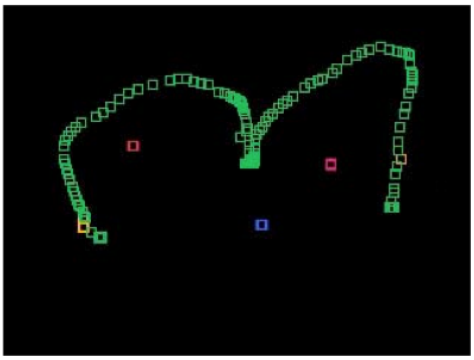
Subjects	Test Environment	Resolution	Frame rate	Number of test scenarios
3 Adults	Lab	320x240	30fps	30 (10 each)
Child #1	Home	640x480	30fps	28
Child #2	Home	320x240	30fps	22
Child #3	Home	720x480	30fps	20

Figure 15: System evaluation in various illumination conditions using different camera resolutions. (Top Left) An adult participant in a controlled lab environment. (Others) Child participants in their home environments.

adults and 83.61% for the children. The sequencing was performed with 100% accuracy based on the play primitives that were recognized. The lower recognition rate in child participants was due to several reasons: The participants became too excited and manipulated the objects outside of the camera frame. The participants sometimes moved more than two objects at once that resulted in an object-tracking error.

The average recognition rate for the sequential play primitives using the left-right hidden Markov model was 88.85% while the repetitive primitives using the cyclic model result had 81.95% recognition. The primitives with the lowest recognition rate were the circular-shaking primitives (78.48%). The training and testing results show

(a)



(b)

Figure 16: Evaluation result of the motion-primitives sequencing for a stacking task. (a) 26th, 66th, 110th, 129th, 186th, and 214th frames of the session. (b) The trajectory of the play object. The sequence was correctly identified as $< \uparrow \leftarrow \swarrow \downarrow$ STACK $\uparrow \nwarrow \leftarrow \swarrow \downarrow \bullet >$

that the behaviors observed during play tasks could be recognized by sequencing low-level play primitives. This fact is very encouraging in a sense that when applied to a robot playmate, the robot could observe and learn the tasks to engage the user in therapeutic tasks.

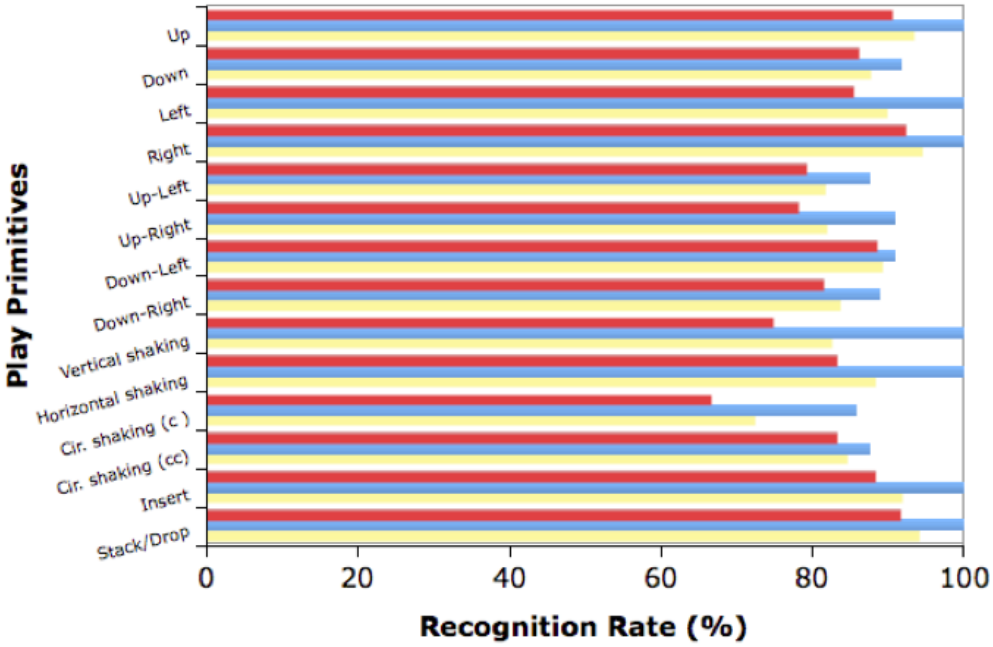


Figure 17: Average recognition rates of each play primitive. The bars indicate the recognition rates for the following participant groups: child(red), adult(blue), and overall(yellow).

		Actual													
		Up	Down	Left	Right	Up-Left	Up-Right	Down-Left	Down-Right	Vert. Shak.	Hor. Shak.	Cir. Shak. (c)	Cir. Shak. (cc)	Insert	Stack/Drop
Predicted	Up	91		1		5	4			2			1		
	Down		73					1	3	1				1	1
	Left	2	2	62		2									
	Right				57		1					1			
	Up-Left	2		2		49									
	Up-Right						35								
	Down-Left			1				49							
	Down-Right								3						
	Vert. Shak.	1								2		2	1		
	Hor. Shak.			1	1						12				
	Cir. Shak. (c)											1			
	Cir. Shak. (cc)				1								12		
	Insert		6					2	3					61	3
	Stack/Drop							1						2	59
	Unidentified	3	3	4	2	5	3	2		1	1			3	
	total	99	84	71	61	61	43	55	36	24	13	13	14	67	63

Figure 18: Play-primitive-recognition result shown in a confusion matrix.

3.4 Online Training of Task Primitives with TabAccess

As was discussed earlier, we can model complex gestural behaviors by sequencing motion primitives. In the previous sections, these primitives were identified by studying the most frequently observed gestural patterns of the task. However, in some tasks, online training of such primitives are important, especially when the task primitives differ depending on the person demonstrating them. This section introduces an assistive device that is built for individuals with limited upper-body mobility to access computers including smartphones and tablets (Figure 19). In the following, a method for training gesture primitives is discussed that allows individual users to customize their device through providing simple demonstrations of the task.

(a) Wearable TabAccess (b) TabAccess Adapter

Figure 19: TabAccess is a controller for computer accessibility including tablets for individuals with upper-body motor impairments.

3.4.1 Motivation

Smart devices are becoming acceptable to a wider population by providing advanced accessibility features in their intuitive touch interface. However, the capacitive touchscreen, which has made smart technology possible, is the same reason some why people are being left out. Over three million individuals in the United States have a disability in their hands and/or forearms and thus have difficulties in effecting pinch

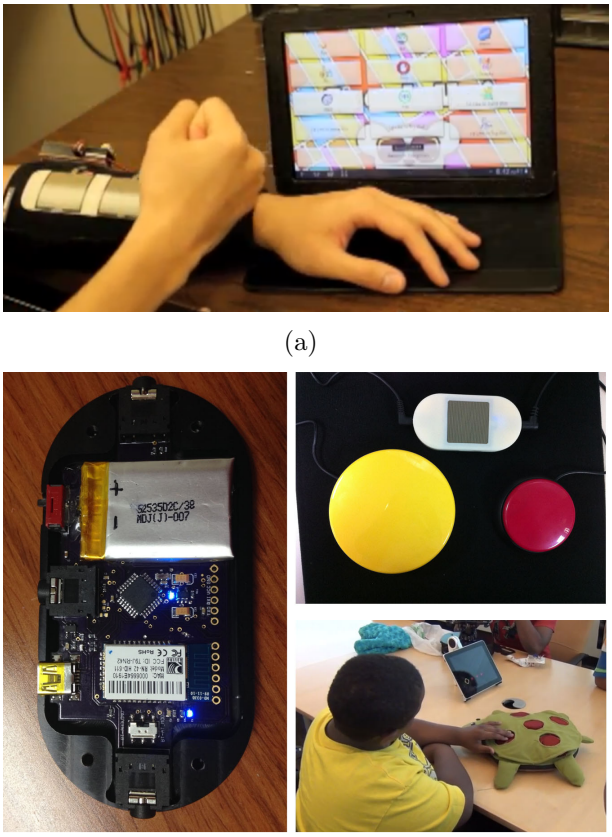
and swipe gestures needed for tablet interaction. TabAccess (controller for Tablet Accessibility) is a wireless controller for individuals with motor impairments designed to provide access to the world through tablet interaction. With the growing availability of smartphones and tablets in our society, individuals are becoming proverbial users that enable them to explore the expanding world of apps, games, and social networks. In fact, numerous articles report how tablet computers are used to engage individuals, with and without disabilities, in a range of cognitive, social, and physical activities by using the tablet's attractive, easy-to-use interface and design. Unfortunately, these touch-based tools are developed assuming the user is capable of 'touching' a specific small region with appropriate intensity and timing. This assumption does not generally hold true when considering individuals who possess limited upper body motor control, such as observed in individuals living with cerebral palsy, Parkinson's, or traumatic brain injury.

In the following, the design of TabAccess is introduced and an online-training of gesture primitives are evaluated. Hidden Markov model (HMM) is trained to recognize different gestures measured by a combination of triggered sensors. Training and testing results are shown with an application developed to play music, drive a robot, and deliver simple conversations.

3.4.2 TabAccess Design

In Figure 20, the designs of TabAccess are shown. The first TabAccess design (Figure 20(a)) is a forearm mountable device designed to slide onto the arm like a sleeve and has three large pressure sensors, which individuals with upper-arm mobility deficiencies can access given their effective range of motion. The device sensors, consisting of force sensitive resistors coupled with an Arduino microprocessor, are placed on an adjustable brace to allow one size to fit the majority of an individual's forearm. For

translation of gross motor gestures into touch-screen based gestures, we have developed a methodology to convert raw sensor data retrieved from the sensors into "press" and "swipe" gestures. While observing users interacting with the device, we found that each individual varied in how they use the device. Some applied force with their large side of the fist, some with their narrower side, some with their hand open, and others using just their index finger. This made us consider implementing training and calibration as an essential part of the device. Figure 21 shows two different subjects performing a *Swipe* (swiping across all the sensors).



(a)

(b)

Figure 20: Feedbacks received from the end-users are reflected in the designs of TabAccess.

In the following section, we describe how the design of TabAccess provides the

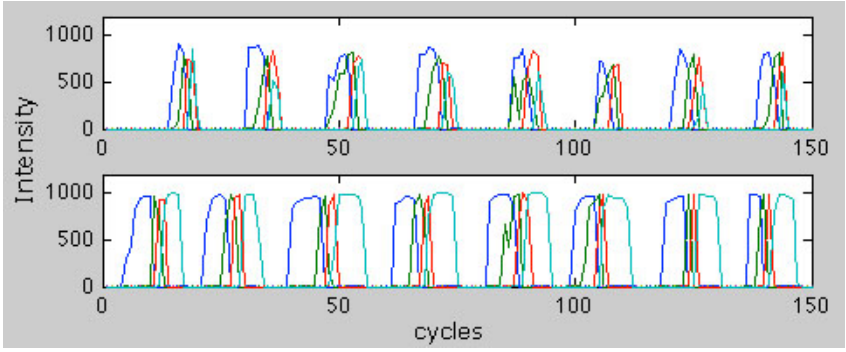


Figure 21: Two subjects performing *Swiping*. Notice not only the intensity but also the duration on each sensor differs.

ability to generate a number of unique gesture commands using the wireless device (i.e. by pressing one of the three resistive force sensors or performing a "forward swipe" or "reverse swipe," which occurs when the user slides their hand, fist, or arm across multiple sensors in either direction). Once generated, the readings from the sensors are transmitted wirelessly to the tablet platform via a Bluetooth connection and decoded by our App interface protocol, which runs in the background and provides input interrupts to any currently active App. Figure 22 diagrams the interaction between the system modules.

The second design of TabAccess (Figure 20(b)) discarded the input surface and attached the connection ports to accommodate commercially available switch devices. The design decisions were made from the feedbacks received from the users who already were using some types of input devices, such as the button switches, sip and puff, head switches, joysticks. This design performs as an adapter that bridges connectivity between mobile devices and switch inputs. The TabAccess adapter provides customization by allowing the user to choose the best switch interface that is accustomed to their ability. Figure 23 shows some available switch adapters that are currently in the market. While Figure 23(a), (b), and (c) provides limited access to

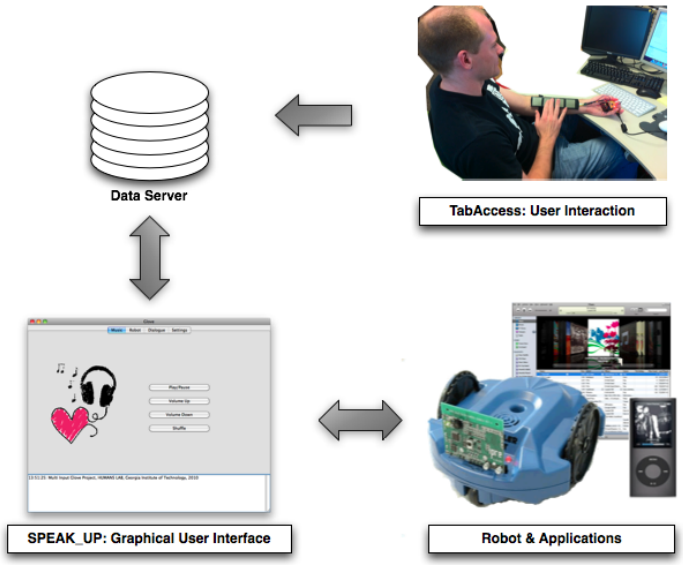


Figure 22: TabAccess system diagram.

custom designed mobile apps, (d), (e), and (f) are designed to provide general navigation to launch apps, but the number of applicable apps are limited. The TabAccess adapter addresses both functionalities and attempts to further increase the adaptability by designing the system to work with different operating systems, i.e., iOS, Android, and Windows.

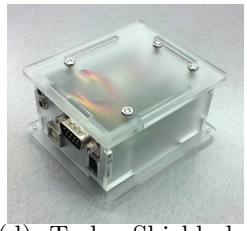
3.4.3 Evaluation and Discussion

Users can create customizable functions by sequencing and combining switch inputs on or connected to TabAccess. During various events, exhibitions, and outreach opportunities held in the past two years, users trained motion primitives with TabAccess and used TabAccess as an input interface to control different functions on the computer and smart devices (Figure 24). Individuals had different motor-impairment conditions, such as cerebral palsy, muscular dystrophy, spina bifida, and quadriplegia. For people already using switch-input devices, e.g, button switch, joystick, and head switch, TabAccess acted as an adapter for accessing computers.

(a) Bluetooth Super Switch by RJ Cooper

(b) Blue2 by Ablenet

(c) Applicator by Pretorian Technologies



(d) Tecla Shield by Komodo OpenLab

(e) Switch2Scan by Pretorian Technologies

(f) VO-Controller by RJ Cooper

Figure 23: iOS and Android smart phone and tablet switch adapters.

For the evaluation presented in this section, six gestures were trained with Hidden Markov Models (HMMs) (Figure 25). The six primitive gestures were: *press and release* of the four buttons, *swiping* through button 1 to button 4, and *reverse-swiping* through button 4 to button 1. These combinations were defined for basic evaluation purposes, but the biggest advantage of using HMMs is that it can be customized to each individual's needs. For example, if a user experiences difficulty swiping through all four sensors, the primitive can be trained to swipe through the last two sensors.

A discrete HMM is represented by three matrices, $\lambda = (A, B, \pi)$. The matrix $A = \{a_{ij}\}$ represents the state transition probability from state i to j. $B = \{b_{jk}\}$ specifies the probability of generating observational symbol k from state j, and π indicates the initial state probability distribution. Before applying raw sensor data to the HMMs, an array of four sensor values were quantized into discrete symbols. The four sensors were classified as either *on* or *off* with a low threshold. The combination of four sensor states generate $2^4 = 16$ codes. After a sequence of sensor value vectors

Figure 24: Users interacting with TabAccess to access smart devices in various events.

$\vec{S} = \vec{S_1}\vec{S_2}\vec{S_3}...$ are converted into discrete observation symbols $\vec{O} = \vec{O_1}\vec{O_2}\vec{O_3}...$, the sequence data is used either to train the HMM $\lambda_i = (A_i, B_i, \pi_i)$ or to recognize which HMM it belongs to by computing the maximum likelihood $arg\max_{\lambda_i}[P(\vec{O}|\lambda_i)]$, where $i = 1...6$. Figure 25 shows a sample input sequence being evaluated.

TabAccess was tested on our applications in Figure 26. *Swiping* and *reverse-swiping* navigates through the applications back and forth. The four button presses are configured to execute different functions on each application. For example, if the user navigates through the menus and chooses to listen to a music, the four button presses serve as play/pause, volume up and down, and shuffle playlist. Such configuration allows us to easily add on more applications.

3.4.3.1 Training

In this work, we trained Hidden Markov Models (HMMs) to recognize six gestures for each subject. The purpose of training HMMs for every user is to calibrate and customize the device to fit each individual's habit and motion range. We collected 250 cycles of data for each of the six gestures per participant. The average sampling rate was 19.62 Hz, and the participants were asked to repeat the same gesture during the 250 cycles of data collection. Figure 27 shows an example of data collection for

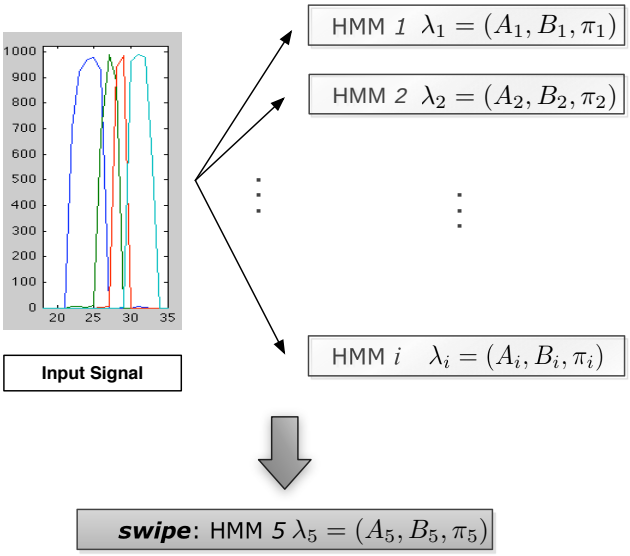


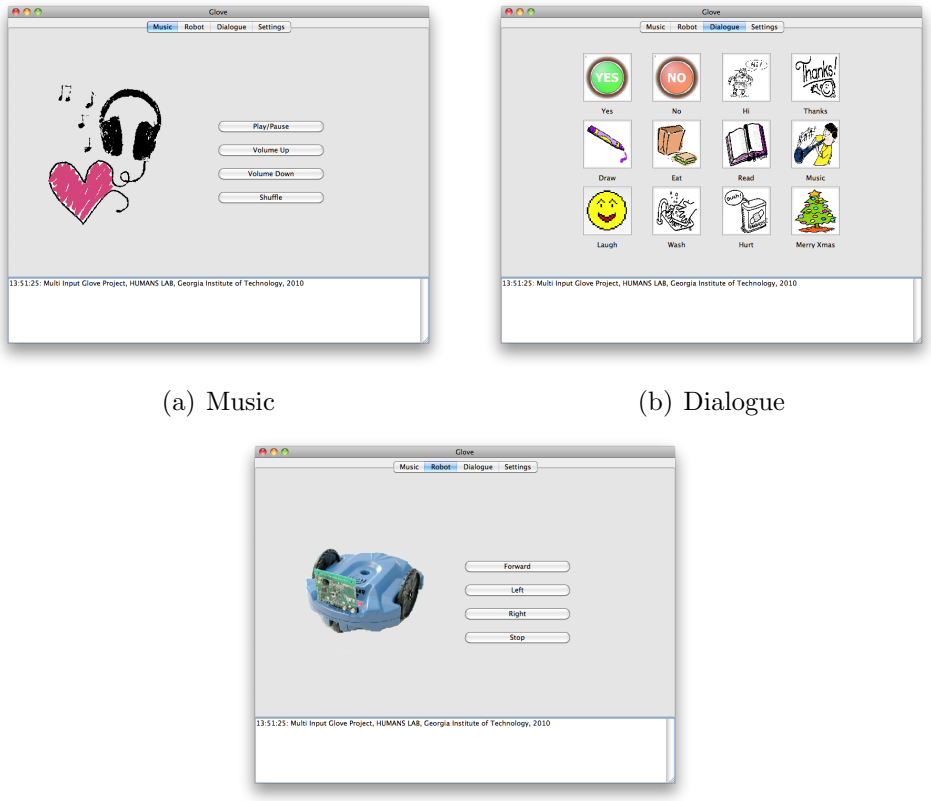
Figure 25: Input signal sequence is evaluated and the HMM with maximum likelihood is selected.

150 cycles. Six users participated in the formal study, and the total training time ranged from 76 to 97 seconds.

3.4.3.2 Testing and Evaluation

Following the training session, a set of testing data was collected in the same manner as the training data. Such collected testing data was used for evaluation, and afterwards the participants freely navigated through our application's graphical user interface and the sub-applications. Figure 28 shows the recognition rate and the confusion matrix. Overall average recognition rate was 96.35%.

The result demonstrates that the gestures generated by different combinations of the sensors can be easily trained and applied to real world applications. This fact is very encouraging in a sense that when used with tablet computers, it adds mobility and grants access to all the features the computer can offer.



(a) Music

(b) Dialogue

(c) Robot

Figure 26: TabAccess applications developed for testing. *Swiping* and *reverse-swiping* navigates through the applications, and *pressing* buttons trigger different event on each application.

3.5 Summary

This chapter presented a research effort on learning a task through extracting and sequencing motion primitives. The objective was to find out how identifying general motion trajectory and the final states of task objects can help the robot's learning. First, the approach was evaluated during children's object-play tasks. A study was conducted to identify the motion segments that comprise general object-based plays. Hidden Markov models were trained to recognize these primitives while observing a child perform a task, and the outputs were sequenced to produce a general trajectory of the task objects.

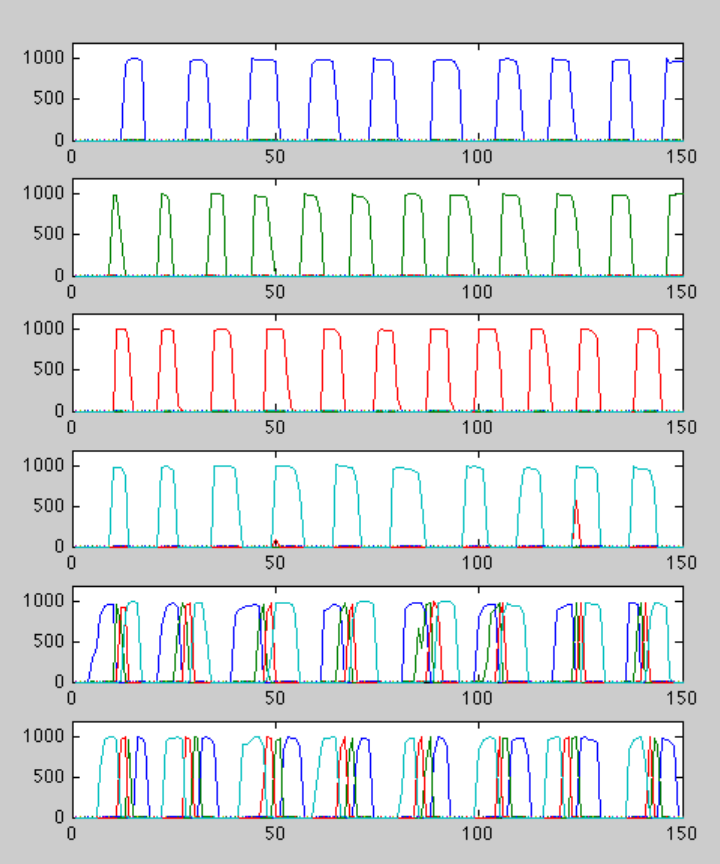


Figure 27: Data Collection (150 cycles) - Top to bottom: Button 1, Button 2, Button 3, Button 4, Swiping, and Reverse-swiping

The second half of the chapter presented TabAccess, a touchscreen tablet computer controller for people with limited upper-body motor functions. Though motion-primitive-based approach to robot learning revealed many issues that in turn motivated the rest of the research in this dissertation, the method inspired an exciting development of a customizable input interface for persons with disabilities. The purpose of TabAccess is to engage individuals in both therapeutic and entertaining interaction with smart devices. The device is either wearable with sensors mounted on a forearm sleeve or acts as an adapter that provides connectivity to traditional switch input devices. With this device, we validated online training of motion primitives. Since each individual's disability is unique, TabAccess is designed to be calibrated

		Observed						Recognition Rate (%)
		Button1	Button2	Button3	Button4	Swipe	Reverse-Swipe	
Predicted	Button1	62				1		98.41%
	Button2		65				1	98.48%
	Button3			68			2	97.14%
	Button4				67			100.00%
	Swipe					53	2	96.36%
	Reverse-Swipe	4			1	3	55	87.30%
		Average						96.35%

Figure 28: Confusion Matrix: predicted versus observed number of test sequences are shown. Six users participated in the study resulting in an average recognition rate of 96.35%

and re-calibrated for each user. During calibration, the gesture primitives were identified and re-defined which were later used to recognize a sequence of commands. TabAccess is currently in the last stage to be commercially available in the market, and we hope that it will make many people's life more enjoyable and accessible.

Though the extraction and recognition of the task motion primitives were successful, for the robot to be able to participate in the task requires more comprehensive information about the task, such as which objects to manipulate, in what order, and what are the object's characteristics. With pattern-recognition techniques or rule-based learning methods, processing this information all together in the system is near impossible. In the following chapter, we discuss how instance-based methods provide an effective learning framework for parsing data that varies in representation, and work as an umbrella term to aid other learning mechanisms. We then introduce interactive learning approaches during human-robot interaction to address the issues of instance-based learning methods.

CHAPTER IV

INTERACTIVE INSTANCE-BASED LEARNING

Interactive instance-based learning (IIBL) attempts to combine two elements among the principal keys to learning: memory and acquiring experience through interaction with others. We believe that, for robots to possess knowledge about multiple tasks, a memory-based approach should be used as an umbrella framework for other learning methods. In this chapter, we describe how interactive and instance-based methods of machine learning aid each other to provide an effective environment for human-robot task learning.

First, case-based reasoning, a representative instance-based method, is briefly introduced along with previous research efforts in using learning from demonstration (LfD) to aid the process of collecting cases. Next, the issues in instance-based learning are presented, followed by discussions on how interactive learning can facilitate the process of instance encoding and acquisition, including the maintenance of the case base. We review two methods for increasing the performance of k-nearest neighbors (k-NN) to compute similarities between instances. k-NN predicts a given query's label using k number of samples that are the most closest to the query [34]. To determine the distance, we adopt regression methods assuming a linear and nonlinear relationships between the task features to recommend a vector of feature weights used towards designing a weighted distance function for k-NNs. In this process, we introduce multivariate locally weighted linear regression and sensitivity analysis using artificial neural networks to measure how much each task feature contributes to maximizing the task objective. Later, an integrated systematic approach is presented that illustrates the interaction between IIBL and these regression modules. At the

end of this chapter, evaluation methods are presented for each of the hypotheses proposed, followed by a description of our robot platform.

4.1 Introduction

4.1.1 Case-based Reasoning and Interactive Learning

Case-based reasoning (CBR) solves new problems by adapting solutions of similar problems experienced in the past [64]. CBR is an instance-based (lazy) learning method where computation is performed at the instance-query time using cases stored in memory, compared to rule-based (eager) methods in which generalization is conducted during model fitting, and data used for the training are discarded [31]. The nature of instance-based methods provide effective task modeling even when prior knowledge of a task is limited, and an explicit model of the problem domain is difficult to elicit, i.e., when a complete mathematical modeling is difficult.

The quality of a case-based reasoner depends on three components: collecting experiences (acquisition), defining the representation of experiences (encoding), and the ability to recall previous experiences (retrieval). In most knowledge-intensive CBR systems, the case base is preloaded before the system deployment [5, 35, 52, 65, 124]. However, recently there have been successful efforts in applying learning from demonstration (LfD) techniques to automate the process of case acquisition within a CBR framework, sometimes referred to as lazy-LfD approaches. A data-driven CBR using crowd-sourcing was presented in [32]. In this work, the proposed system collected instances during human-human collaborative task in a virtual reality environment. Afterwards, the case base was used towards a similar task conducted in the physical world with a human-robot team to generate robot behavior. In [87, 89], the authors have solved the issue of populating case base with plans through LfD for generating planners for real-time strategic games.

Based on the success of previous works, we propose to implement LfD in an

interactive learning setting in which the human teacher provides demonstrations face-to-face to a robot learner in a shared workspace. Unlike most CBR implementations in which they separate the process of populating the case base prior to application (batch learning), IIBL's case acquisition happens at the same time the user and the robot carries out the task. The benefit of such setting is in that the user monitors the robot's performance in real time and interactively provides necessary demonstrations when needed. Such interactive learning methods combining the structure of the LfD were reported to improve both the learning performance of the system as well as the teacher's experience [91, 93, 132].

4.1.2 Issues of Instance-based Learning

Most CBR implementations, including the above mentioned works, rely on the expert to represent and retrieve instances. Since CBR relies on accumulated knowledge, it would be possible to build a robot that stores multiple knowledge libraries linked to different tasks. However, there are several hindrances to automating this approach even if we overlook the problem of handling large data. These problems are associated with the issues of acquisition, encoding, and retrieval of task instances, and our approach is summarized in the following:

- ***Acquisition*** is the problem of how cases are collected. Our robot learner acquires cases through interaction with the teacher who demonstrates the task on a shared workspace. The robot extracts task states from the scene and associates the states with the teacher's behavior that produces a change in the task states. Since CBR and other lazy learning methods respond to a given query by combining information from stored data, the quality of retrieved cases depends on how well the system's case base spans the task space. Though there are various ways to measure the quality of retrieval results, querying the system for better cases around that query instance, especially in real time, is a complex

problem. The instances should cover the goals and subgoals for the task, and the case base should include both successful and unsuccessful attempts toward those goals. Unsuccessful instances are used to avoid the solution and inform the teacher to update the case base with better experiences. With our proposed approach of providing demonstrations through natural interaction in a shared workspace, the teacher intuitively monitors the progress of the robot learner in real time. Therefore, the teacher is able to interrupt the robot's behavior and provide necessary cases at the moment learning is taking place, thus providing a means to continuously engage the participant in the task.

- ***Encoding*** is the problem of case representation. Any CBR system requires to be seeded with a representation of experiences. The challenges of feature selection during human-robot interaction have been studied by imitation-learning researchers as the issue of "what to imitate" [16, 19, 84]. Attention systems such as saliency-based visual attention that integrates face and gaze tracking, emotion detection, gesture recognition, and shared task awareness have been explored. Unfortunately, when it is difficult to acquire a full view of the user and the task scene, or when an alternative method of tracking such cues is unavailable, the aforementioned methods are hardly applicable. Instead, we develop an interface for the users to input task-feature properties that they think are relevant to the task. The significance of each feature variable is determined during training a retrieval metric. Such a method of human intervention is widely used to mediate uncertainties in the environment.

- ***Retrieval*** is the problem of computing a similarity measure for finding nearest

case instances. We adopt a linear regression method for computing a case-retrieval function with little or no domain-specific knowledge. Since our goal is to use a domain-independent framework to model tasks with different types of features, we convert the feature space into a real-valued feature-distance space and apply locally weighted regression (LWR). Basic LWR is a derivative of the k-nearest neighbor (k-NN) classifier and has been implemented extensively throughout CBR based applications. Aha [127] and Wettschereck [126] provides a review on several weighting methods for lazy-learning algorithms based on dataset characteristics. Among the methods reviewed, the authors point out that the algorithms using performance feedback require less pre-processing, better tolerate interacting features, and increase the learning rate. Atkeson et. al [7] and Schaal et. al [106] provide introduction to LWR and its variations, and present examples of applying the algorithm to learn to balance the swing-up pendulum. However, since linear regression only assumes linear relationship between independent variables, there was a need to explore other methods that would model the nonlinear behavior of the dependent features. An approach using artificial neural networks (ANN) was evaluated that determines the contribution of each feature variable using sensitivity analysis [42, 44, 73, 104]. Such approach adds power to ANN in their explanatory capacity, and this dissertation attempts to further boost the transparency of learning by developing a hybrid approach to IIBL with neural networks.

4.2 Approach

In Figure 29, the overall flow of the IIBL framework is depicted. IIBL framework consists of instance-based learning processes that is supplemented by the interactive learning methods. In this dissertation, we address the issues of acquiring, encoding, and retrieving cases, and the approaches we took are introduced in the remainder of

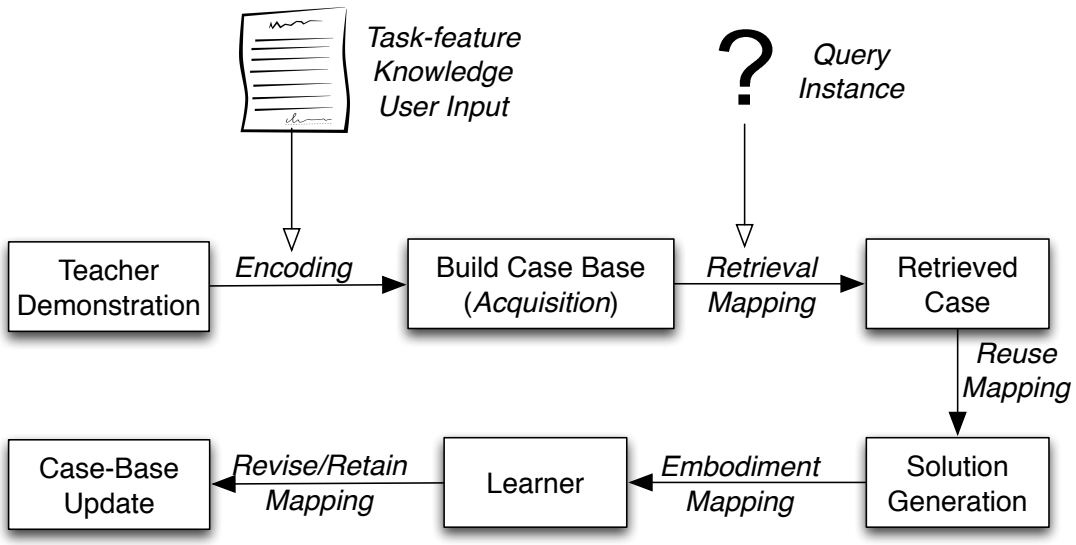


Figure 29: The flow of the IIBL framework that includes recording and encoding demonstrations, retrieving and reusing cases, and mapping a generated behavior to the robot's embodiment.

this section.

4.2.1 Acquisition

An instance is defined as a tuple composed of a problem and a solution. When demonstrations are recorded, the current state of the task is encoded as a problem. Later when a new query is given, the problem portion of an instance is compared to the query to compute the distances between instances. A solution is recorded as the teacher's behavior that influences the change in the task states. Instance acquisition in IIBL usually involves robot's perceptual capability, so it is important for the teacher to be informed about the robot's skills before encoding task features. For example, during a block stacking and inserting task in Chapter 5, the task features are extracted using visual information, and the problem and solution are formulated as a collection of objects' attributes and the teacher's general motion trajectory that were involved

in a turn. In Chapter 6, the robot learns a task on the tablet from the teacher, and the user's response to that state on the tablet is encoded as the solution. The robot acquires the information of task states from the tablet through a UDP socket communication, from which state packets are sent to the robot in some sampling rate. When the user initiates any touch event on the tablet, the start and end coordinates are sent to the robot; the robot then knows that a demonstration was given and creates a case. In the same way, the robot sends the start and end coordinates of a synthesized touch event to the tablet and computes inverse kinematics for its head and arm joints to generate a hand-eye coordinated motion.

The issue of acquisition is not only limited to accumulating new instances. It helps the task domain to be better covered with successful and unsuccessful instances through the interaction with the teacher and the workspace shared among them. After the robot learner produces a prediction of its behavior, the teacher's response triggers the system whether such behavior was successful or not. If the behavior was considered unsuccessful, the previous instances that generated the solution are updated with the correct behavior the teacher provides.

4.2.2 Encoding

A case is a 2-tuple model:

$$C = \{D_{prob}, D_{sol}\}$$

where D_{prob} is a problem descriptor and D_{sol} is a solution descriptor. The problem and solution descriptors are consisted of task features:

$$D_{prob} = \{f_1^p, f_2^p, \cdots, f_n^p\},$$

$$D_{sol} = \{f_1^s, f_2^s, \cdots, f_m^s\}$$

where n and m are the numbers of the problem features, f^p, and the solution features, f^s. Domain-dependent feature descriptors are:

$$f_i^p : \{x_i^p, attr_i^p\}, \text{ where } attr_i^p : \{T_i^p, M_{ex_i}^p, M_{dist_i}^p, w_i\},$$

$$\text{and } f_j^s : \{x_j^s, attr_j^s\}, \text{ where } attr_j^s : \{T_j^s, M_{ex_j}^s\}.$$

The feature space variables including the feature value x and feature attributes $attr$ are:

1. x: the feature value

   The feature *value* x of data type T is extracted with the method M_{ex}. A similarity between the two feature values is calculated by the distance function M_{dist}, and the resulting similarity measure influences the overall case similarity by the factor of w. Generally, you cannot express data with different types into a vector, but for convenience in representation, we will henceforth write problem features as $\mathbf{x}^p = \{x_1^p, x_2^p, \cdots, x_n^p\}^T$.

2. T: the feature data type

   The feature *data type* in CBR could be in many different forms including string, integer, boolean, float, and vectors of these data types. The IIBL system supports the following programming data types:

   $$T \in \{string, int, float, vector < string >, vector < int >, vector < float >\}$$

3. M_{ex}: the feature extraction method

   $$template\ M_{ex} < T >:\ T\ M_{ex}\,(\,T\ var\,)$$

   The feature *extraction method* returns x of data type T. The modules first supported visual data processing for object-related activities in a therapeutic setting, such as recognition of object shapes, relative sizes, colors, locations, and manipulation-trajectory primitives. For tablet-based applications, this indicates the method of how to parse data packets sent from the tablet. The function takes an optional variable var.

4. M_{dist}: feature distance-measure method

$$template\ M_{dist} < T >:\ float\ M_{dist}\,(T\ \mathbf{p}_a,\ T\ \mathbf{p}_b,\ T\ var\,)$$

The feature *distance metric* measures the distance between two feature values of data type T and returns a float value. This results in a real numeric value for all feature types and can now be represented in an n-dimensional space. Popular similarity measures such as logical AND and OR operators, min/max, Euclidean, mean, and median are supported. The returned float value is normalized to $\in [0, 1]$. The function takes an optional variable var.

5. w: feature weight

Individual feature distances are weighted according to their relative contribution to the overall distance between instances, and the weights are trained using regression methods. The first process is explained in detail in the next section, where a locally weighted regression (LWR) method is used with feature-distances as an input space. The coefficients are specified such that they minimize the squared error summed over the nearest instances of the query feature-distance vector. The LWR's target function doubles as the global retrieval function. The second method uses a sensitivity analysis with neural networks to measure the relationship between an input feature variable and the output variable. This method either removes or changes the feature variable successively to measure the influence in the output. Thorough algorithms are introduced in Section 4.3. The characteristics of the normalized weight coefficients can be summarized as follows:

$$\mathbf{w} = \{w_1, w_2, \cdots, w_n\},$$

$$\text{where}\quad w \in [0, 1] \quad \text{and} \quad |\mathbf{w}| = 1.$$

The system receives the above feature-attribute information from the user through a simple Extensible Markup Language (XML) interface (Listing 4.1). For example, in our evaluation mobile app in Chapter 6, the feature "enemylocation" is a set of (x, y) float vectors that describe the remaining enemies' locations in a game. A metric we use to compute the distance between enemy locations of two problems is "MinVectorAvg" which averages over minimum distances between enemies in two cases. Some of these information are completed by the framework. The framework provides predefined method modules for feature extraction and distance measures.

```
<Problem>
                    ⋮
  <Feature>
    <Value> enemylocation </Value>
    <Def> Array of (x, y) enemy locations </Def>
    <Type> vector:float </Type>
    <Mext> ParseDataPacket </Mext>
    <Var1> 4 </Var1>
    <Mdist> MinVectorAvg </Mdist>
    <Var2> 40000 </Var2>
    <Weight> 0.4500 </Weight>
  </Feature>
                    ⋮
</Problem>
<Solution>
  <Feature>
    <Value> xyTouch </Value>
    <Def> Array of (x,y) coordinate of the touch event </Def>
    <Type> vector:int </Type>
    <Mext> ParseDataPacket </Mext>
    <Var1> 1 </Var1>
  </Feature>
                    ⋮

</Solution>
```

Listing 4.1: Task feature parameters are defined using the provided extraction/distance-metric modules.

4.2.3 Retrieval

The retrieval stage is where the current problem states are compared to the problems of the cases in the case-base. The retrieval function is modeled as a linear sum of locally weighted task features. The weights are trained such that the overall function minimizes the cost function. This approach is similar to maximizing a reward function that penalizes deviations from a demonstrated motion trajectory for solving the swing-up inverted pendulum task [6].

Linear regression is a problem of fitting a linear function to a set of input-output pairs given a set of training examples, in which the input and output features are numeric. The distances between the feature pairs become the input variables:

$$\mathbf{d} = \{\delta(x^p_{1_i}, x^p_{1_j}), \delta(x^p_{2_i}, x^p_{2_j}), \cdots, \delta(x^p_{n_i}, x^p_{n_j})\}^T,$$

where $x^p_{k_i}$ is the k-th feature, and $\delta(x^p_{k_i}, x^p_{k_j})$ is the output of M_{dist_k}. The distance $\delta(x^p_{k_i}, x^p_{k_j})$ will be abbreviated as δ^k_{ij} for simplicity. The target function models a retrieval function assuming a general linear relationship of the feature distances:

$$g(\mathbf{w}, \mathbf{d}) = \sum_{k=0}^{n} w_k \cdot \delta^k_{ij}$$

where $\mathbf{w} = \{w_0, w_1, \cdots, w_n\}$ is the regression coefficient vector, and $\delta^0_{ij} = 1$. A set $\mathbf{E}$ is defined as nearest-neighbor instances corresponding to $\mathbf{d}_q$. The regression coefficient vector $\mathbf{w}$ is then specified in order to minimize the squared error summed over the set $\mathbf{E}$.

$$Error(\mathbf{w}, \mathbf{d}) = \frac{1}{2} \sum_{\mathbf{d} \in \mathbf{E}} (g(\mathbf{d}) - \hat{g}(\mathbf{w}, \mathbf{d}))^2.$$

The gradient descent method is then used to compute $\mathbf{w}$ iteratively. This overall process is a locally weighted regression (LWR) and is a representative method of instance-based learning approaches, except that here, we have applied LWR in the feature-distance space instead of the feature space itself. This process is repeated for some number of query points, and for each query point the nearest neighbor set $\mathbf{E}$ is restated. Note that after training, the target function $g(\mathbf{w}, \mathbf{d})$ is used as the global similarity measure for retrieving cases.

In the next section, the second regression approach to training the feature weights is presented that relies on a sensitivity analysis with neural networks. We attempt to address the limitation linear regression methods possess — their incapacity to model nonlinear behavior of the dependent variables. Throughout Chapter 5 and Chapter 6, we evaluate and compare the performances of the two regression approaches and k-NN.

4.3 Feature-weight Predictions using Sensitivity Analysis with a Feedforward Neural Network

Here, we are proposing a hybrid approach to IIBL with an artificial neural network (ANN). The hybrid system utilizes neural network sensitivity analysis (NNSA) for recommending a feature weight vector $\mathbf{w}$ for IIBL. ANN is a powerful supervised learning method for solving classification and regression problems [73, 103, 113, 131]. A multilayer feedforward neural network can represent a broad set of nonlinear functions. However, as mentioned earlier in Chapter 2, rule-based learning methods, including ANN, provide little comprehensible knowledge about how it arrives at a given result. Discarding the dataset used for training makes the system difficult to trace back to the instances that contributed to providing the prediction. This also makes the system less flexible. When the prediction is far off, instance-based methods can replace or update the instances that caused these results and re-train the system when necessary. In human-robot interactive learning environments, knowing

which demonstrations led to which robot behavior can provide persuasive information for explaining the robot's decision. This encourages the human teacher to provide supplemental or more accurate examples.

To address the shortcomings of ANN, researchers have made efforts in extracting the knowledge embedded within trained neural networks [50, 76, 83, 118]. Here, we propose a hybrid system of instance-based learning and ANN. The instance-based nature of the system complements the explanatory behavior of ANN, and the neural network provides a better estimate of the retrieval mechanism. We have thus far seen that the k-nearest neighbor (k-NN) and its variants are widely used to retrieve cases from memory. k-NN assumes that all input features are equally important, which is acceptable if we know for a fact that the selected features are indeed crucial in modeling a certain task. Since IIBL receives information about the task features from the users, the system requires a mechanism similar to feature selection to assign larger weights to more relevant features and smaller weights to less relevant ones. Previously, we have discussed adaptive feature-weighting methods using locally weighted regression. Here, we will discuss different sensitivity-analysis approaches using a trained feedforward neural network.

The goal of the sensitivity analysis is to evaluate the relative importance of the input features. It measures to what degree each input feature contributes to the change in the prediction result. The information obtained from the sensitivity analysis also can tell the form of relationship between the variables, such as x_i is positively related to y_k. Some analysis can also report the form of relationship in the context of other variables.

First, let's assume a trained neural network in Figure 30 with n_I inputs x, one hidden layer with n_H neurons z, and n_O outputs y. In the figure, the black lines are positive weights and the grey lines are negative weights while the thickness of the lines is in proportion to the magnitude of the weight relative to others.

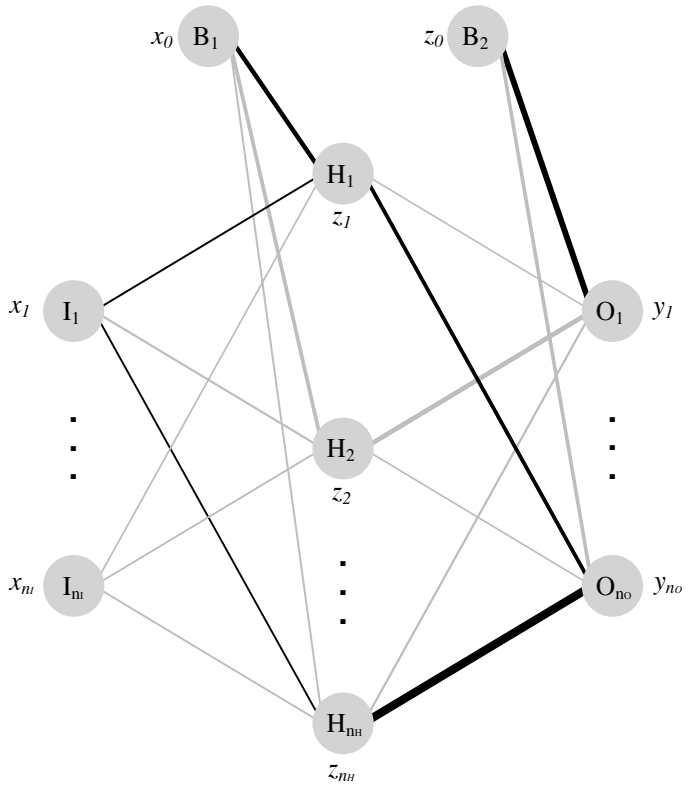


Figure 30: A fully connected, feedforward neural network with one hidden layer.

The input of the j-th hidden neuron, a_j, is obtained by first forming a weighted linear combination of the n_I input values and then adding a bias. The input of the k-th output unit b_k is obtained in the same manner.

$$a_j = \sum_{i=0}^{n_I} v_{ji}^{(1)} x_i \tag{20}$$

$$b_k = \sum_{j=0}^{n_H} v_j^{(2)} z_j \tag{21}$$

Here, $v_{ji}^{(1)}$ denotes a connecting weight from x_i to z_j, and $v_{kj}^{(2)}$ denotes a connecting weight from z_j to y_k. The bias terms of hidden and output layers ayer treated by the inclusion of extra variable $x_0 = 1$ and $z_0 = 1$. The activation of z_j and y_k is then

obtained by transforming the linear sum in Eq. 20 and Eq. 21 using an activation function. In biologically inspired neural networks, the activation function is an abstract representation of the rate of action potential firing in a cell. Neural network requires a differentiable activation function to calculate backpropagation. A popular form of the function is a sigmoidal activation function, $\phi(\alpha) = (1 + exp(-\alpha))^{-1}$. An explicit expression of the output function in Figure 30 is then:

$$y_k = \phi\left(\sum_{j=0}^{n_H} v_j^{(2)} \phi\left(\sum_{i=0}^{n_I} v_{ji}^{(1)} x_i\right)\right). \tag{22}$$

The backpropagation technique modifies the connection weights to minimize the error of the prediction. The training of the network continues until the sum of the squares of errors (SSE) is minimized:

$$SSE = \frac{1}{2} \sum_{\forall observations} (Y - \hat{Y})^2 \tag{23}$$

where Y is the expected value of the response variable and $\hat{Y}$ is the value obtained from the network. Refer to [54] for more information on backpropagation-based network training.

As was in the previous section, the goal is to obtain the weight vector $\mathbf{w} = \{w_1, w_2, \cdots, w_n\}$, where n is the number of task features. After reviewing the literature, we have selected three approaches to computing the feature weights using neural networks. The obtained weight vector is then used to retrieve k-nearest neighbors in the case base. The prediction results from the neural network may also be utilized to aid the final solution of the IIBL framework.

For evaluation purposes, a neural network with five inputs ($n_I = 5$), one hidden layer with four neurons ($n_H = 4$), and one output node ($n_O = 1$) was trained with a training dataset. The input variables were generated using a multivariate normal

distribution $\mathbf{x} \sim \mathcal{N}_{n_I}(\mu_x, \Sigma)$ with $\mu_x = \{\mathbf{0}\}$ and covariance matrix Σ:

$$\Sigma = \begin{bmatrix} 8.5320026 & -1.9609842 & 0.1546096 & 3.2196470 & -0.5332321 \\ -1.9609842 & 2.3545130 & 0.2903274 & 0.9781670 & -0.3093482 \\ 0.1546096 & 0.2903274 & 4.1254502 & -2.5462860 & -0.2845803 \\ 3.2196470 & 0.9781670 & -2.5462860 & 5.3989591 & -0.1440045 \\ -0.5332321 & -0.3093482 & -0.2845803 & -0.1440045 & 2.3432218 \end{bmatrix}, \quad (24)$$

and the output variable followed a normal distribution with some noise introduced to the equation:

$$y = H\mathbf{x} + \eta \quad (25)$$

$$= \begin{bmatrix} -9.964524 & 6.700290 & 3.029034 & 4.489175 & -7.844071 \end{bmatrix} \mathbf{x} + \mathcal{N}(\mu_y, \sigma_y^2), \quad (26)$$

where the values were set to $\mu_y = 0$, and $\sigma_y = 20$. A graphical representation of the network is depicted in Figure 31.

4.3.1 Shin's Sensitivity Measure

In [111], Shin et al. adopted methods of neural network pruning for feature weighting. Network pruning is a practical method to minimize the size of the network, while maintaining good performance [98, 107, 108, 125]. In this literature, the sensitivity of each feature is calculated by removing the input node from the trained neural network. Instead of physically removing the neuron and connected synapses, the weights connected to the input neuron are set to zero. Then the sensitivity is measured by the difference in the prediction result when the feature is present and when it is removed. The sensitivity S_i of a given input feature x_i is:

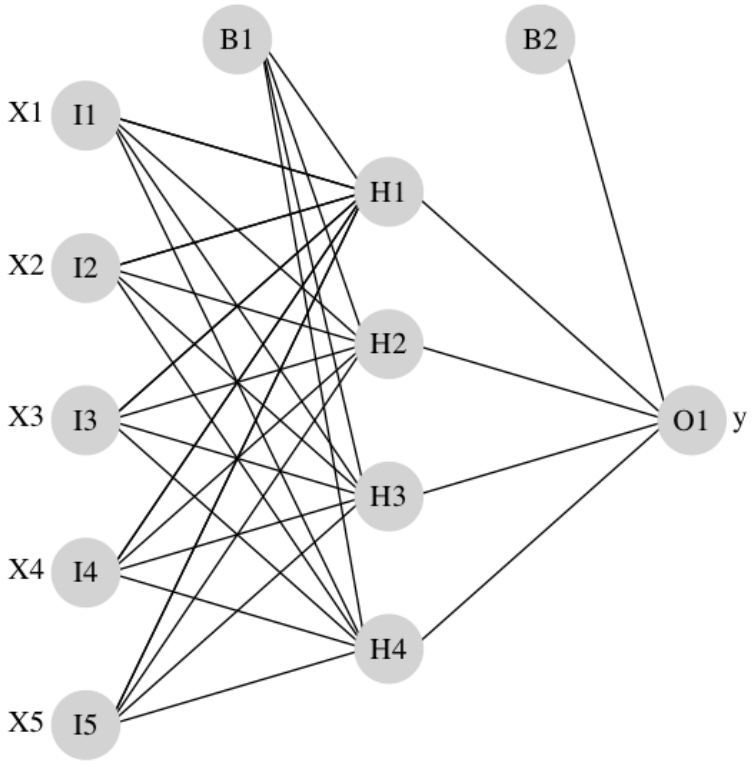


Figure 31: A neural network with five inputs ($n_I = 5$), one hidden layer with four neurons ($n_H = 4$), and one output node ($n_O = 1$) is used for evaluations of sensitivity analysis.

$$S_i = \frac{\sum_{\forall observations} \frac{|P - \hat{P}|}{P}}{N} \tag{27}$$

where P is the normal prediction value for each training instance after training, and $\hat{P}$ is the modified prediction value when the input node i is removed. N is the number of training datasets. In IIBL, we compute P as the system performance of the retrieved solution after the initial training, and $\hat{P}$ as the system performance after the input node i has been removed. Afterwards, feature weights w_i are assigned proportion to their sensitivity value:

$$w_i = \frac{S_i}{\sum_{m=1}^{n_I} S_m}. \tag{28}$$

Table 1 presents the evaluation result using the network in Figure 31. The result shows that removing the input variables I_1 and I_2 produced the most fluctuation in the prediction result. This method reports the relative degree of the influence each input feature possesses in producing the output, but does not present the form of relationship between the variables.

Table 1: Sensitivity measure and weight assignment with leave-one-feature-out method

	I1	I2	I3	I4	I5
S_i	1.568749	1.111705	0.900696	0.752329	0.996918
w_i	0.294577	0.208754	0.169131	0.141271	0.187199

4.3.2 Garson's Weights Method

This method proposed by Garson [42] provides the proportional explanation about the importance or distribution of all output weights attributable to the given input variable. First, for each hidden neuron z_j, multiply the absolute value of the hidden-output layer connection weight by the absolute value of the hidden-input layer connection weight. Do this for each input variable x_i to produce a mid-product R_{ji}.

$$R_{ji} = |v_{ji}^{(1)}| \cdot |v_{kj}^{(2)}| \tag{29}$$

Next, for each hidden neuron z_j, divide R_{ji} by the sum of all the input variables to obtain Q_{ji}.

$$Q_{ji} = \frac{R_{ji}}{\sum_{l=1}^{n_I} R_{jl}} \tag{30}$$

For each input neuron x_i, sum Q_{ji} over all j to produce S_i.

$$S_i = \sum_{j=1}^{n_H} Q_{ji} \tag{31}$$

Finally, divide S_i by the sum for all S for normalization. This provides the proportional explanation about the importance or distribution of all output weights attributable to the given input variable.

$$w_i = \frac{S_i}{\sum_{m=1}^{n_I} S_m} \tag{32}$$

For the detailed procedure of the method, refer to [42, 46] and [44]. Considering the previous neural network example with five inputs ($n_I = 5$), one hidden layer with four neurons ($n_H = 4$), and one output ($n_O = 1$) with weight values in Table 2, the following steps were taken to compute $\mathbf{w}$:

Table 2: Weight values for neural network in Figure 31

	I1	I2	I3	I4	I5	Y1
H1	-0.008489	0.443718	0.034606	-0.046344	-0.225199	0.864951
H2	0.118378	-0.146803	0.105522	-0.051689	0.094474	-0.221044
H3	0.097190	-0.053521	-0.150281	-0.099809	0.145209	-0.788355
H4	0.099682	-0.057968	0.047918	-0.059545	0.045585	-0.351483

Table 3 shows R_{ji} computation, and Table 4 shows Q_{ji} computation and the final results of the sensitivity measure and weight assignments.

Table 3: $R_{ji} = |v_{ji}^{(1)}| \cdot |v_{kj}^{(2)}|$ values computed for each neurons in the hidden-input layer.

	I1	I2	I3	I4	I5
H1	0.007343	0.383794	0.029933	0.040085	0.194786
H2	0.026167	0.032450	0.023325	0.011426	0.020883
H3	0.076620	0.042193	0.118475	0.078685	0.114476
H4	0.035036	0.020375	0.016842	0.020929	0.016022

Table 4: $Q_{ji} = \dfrac{R_{ji}}{\sum_{l=1}^{n_I} R_{jl}}$ values, and sensitivity measure and weight assignment with the Garson-weights method.

	I1	I2	I3	I4	I5
H1	0.011194	0.585104	0.045633	0.061111	0.296957
H2	0.229031	0.284025	0.204157	0.100005	0.182782
H3	0.178001	0.098022	0.275235	0.182797	0.265946
H4	0.320832	0.186572	0.154228	0.191649	0.146718
S_i	0.739058	1.153723	0.679254	0.535562	0.892403
w_i	0.184765	0.288431	0.169813	0.133890	0.223101

The Garson's weights method suggests that I_2 and I_5 are the most contributing features, followed by I_1, I_3, and I_4. This method also does not inform the form of relationship between the variables.

4.3.3 Lek's Profile Method

The general idea of this analysis, proposed by Lek [73] and repeated by Gevrey [44], is to study each input feature successively when other input variables are assigned fixed values. This is contrary to the leave-one-out method discussed earlier, in which the input node in question and its connection weights were removed to measure the relative sensitivity. The method assumes that the range (minimum and maximum) of each input variable is known or can be obtained. Then for each input variable, we compute the predictions of each output variable across the range of the variable of interest. All other input features are held at their equally divided intervals, starting from their minimum, then successively at their 20-th percentile until the maximum.

While the variable in question is sequenced across its range, other variables are held constant at their median. The final product is a set of profiles of the output-variable trends that are plotted against the increase of one input variable while other variables are held constant.

Figure 32 depicts the profiles generated for the neural network in Figure 31. For this example dataset, in which there exist some correlation among the input variables, the results illustrate almost linear trend lines, and one can acquire the form of input-output relationship for each input variable. Gevrey [44] proposes to compute the relative contributions of each input variable through the range of the output variable, i.e. $max(Y) - min(Y)$. Table 5 presents the average range of the output variable for each input feature and the corresponding normalized **w** vector. With this method, it was concluded that the features I_1 and I_5 were the most influential variables.

Table 5: Sensitivity measure and weight assignment of the Lek-profile Method.

	I1	I2	I3	I4	I5
S_i	0.552329	0.373125	0.289300	0.385937	0.427406
w_i	0.272339	0.183978	0.142646	0.190295	0.210742

4.3.4 Discussion

In this section, we reviewed three sensitivity analysis methods for measuring relative contributions of the input features toward producing predictions. All three methods were able to provide the order of importance while the Lek-profile method was also able to generate an estimate of the form of input-output relationships. The observations used to build the network in Figure 31 were divided into 10 groups to calculate the average contributions of the input variables and their confidence intervals in Figure 33.

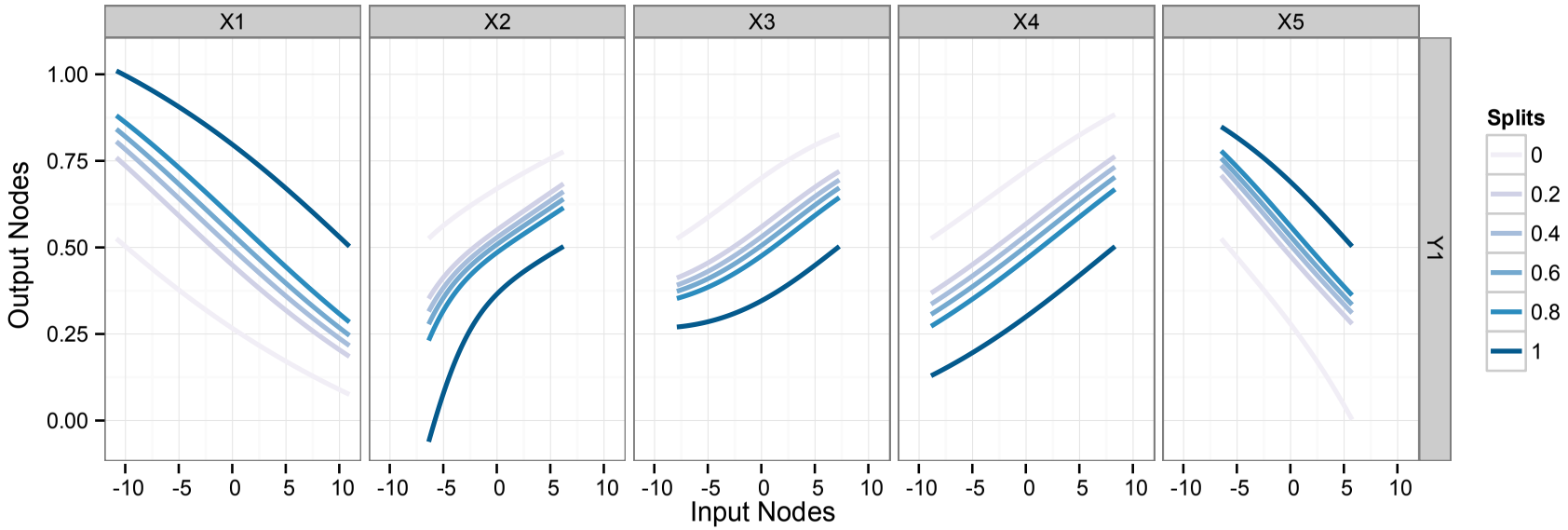


Figure 32: Contribution of each input variables computed by the Lek-profile method.

The figure implies that, in regards to the ordering of the variable contributions, the results observed for each method are not always the same. It is obvious that the difference in computation and normalization methods leads to different results. The confidence intervals plotted for each method indicate their stability. The Lek-profile method provided the most stable results while Garson's weights method was the least stable. Shin's leave-one-out sensitivity measure was the closest result to the average contributions of all three methods.

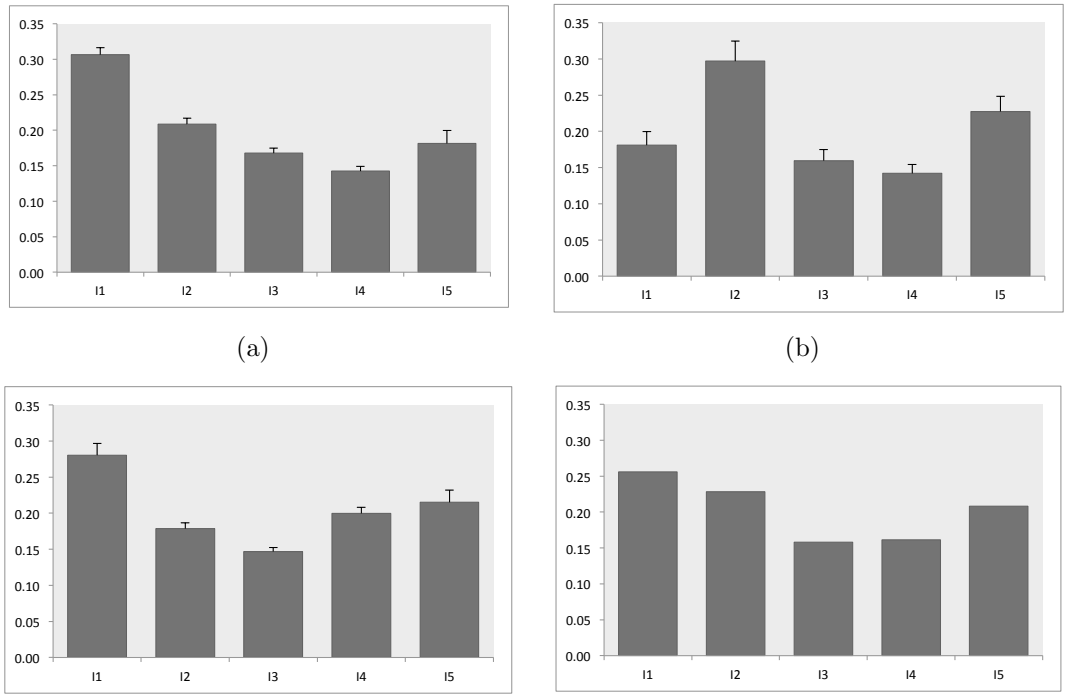


Figure 33: Contribution of input features in each method. (a) Shin's sensitivity measure, (b) Garson's weights method, (c) Lek's profile method, and (d) an average of the three methods.

4.4 A Hybrid Approach to IIBL with Neural Networks

Shin et al. [111] suggested utilizing ANN's outputs as solutions to given problems as well as using the network as a feature-weighting mechanism for instance retrieval. The basic idea is to compare the solution generated by ANN and the solution retrieved

from the case base using the weights ANN suggested to the system. If the solutions match in a classification problem (or the difference is within a threshold in a regression problem), then the system reports the result, and if not, both solutions are rejected.

In IIBL, a neural network is solely used for training the feature weights (Figure 34). Since an IIBL framework is targeting the generalization of task modeling, providing absolute comparison between an ANN generated solution and an instance-based retrieved solution across various tasks is impossible. It would be possible to find a comparison threshold value that caps the rejection rate under a limit through iteration. However, instead of rejecting possible solutions, IIBL prefers to make mistakes that encourages the human user to observe the robot behavior and provide better demonstrations at the right timing and improve the case base. If the frequency of the user providing demonstration increases, ANN re-generates the feature weights with the current case base.

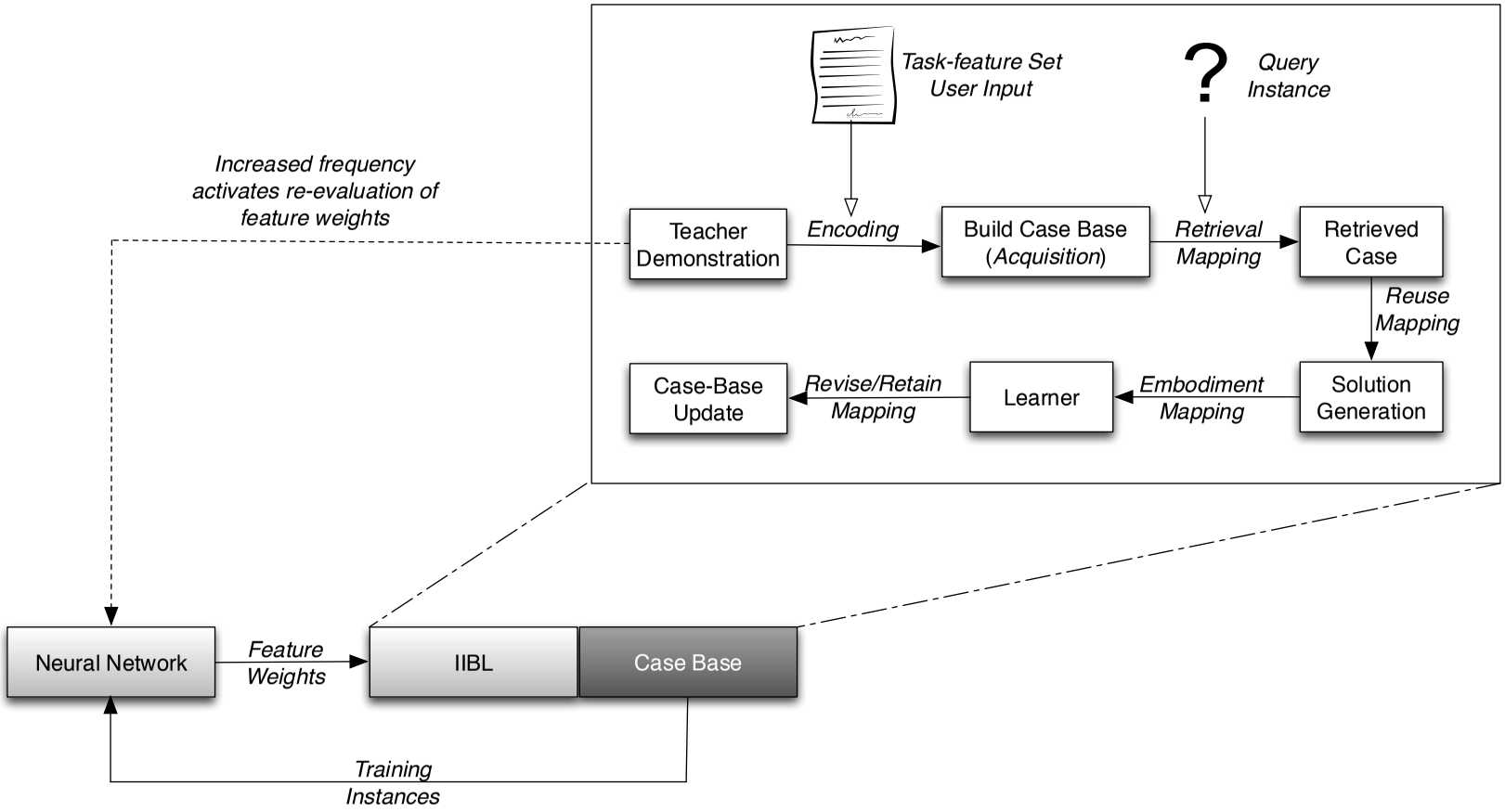


Figure 34: A hybrid approach to IIBL with neural networks system overview. A feedforward NN is generated using initial training instances in the case base. The NN supplies feature weights to IIBL obtained through analyzing sensitivity of each task feature. IIBL retrieves k-nearest solutions using the weights.

4.5 Evaluation Methods

The experiment is conducted to test the following hypotheses.

1) *IIBL provides comparable task performance against the average performance of the demonstrator*: Given a task T demonstrated by a teacher, we evaluate the learned result in three viewpoints as outlined in [89].

   - Evaluating the performance by measuring how well an agent A performs T. For example, a game score can be used to evaluate the performance of Darwin executing the game learned from a human teacher.
   - Evaluating the behavior by comparing the actions executed by A while performing T against the teacher's actions when executing the same task T. For example, measuring the performance of block stacking (Chapter 5) may not be trivial, and the traces of procedure should be compared to the task performed by the teacher. Instance-based learning methods provide the best approach to this problem since it retains demonstrations formulated as cases, whereas many other supervised learning techniques discard demonstrated information once trained.
   - Evaluating the model by comparing the task model generated by A against the model the teacher used to provide demonstrations. Early works on LfD used this method towards tasks with known explicit models for evaluating whether their proposed learning method could recover the given task model [6, 9]. While different from tasks with explicit mathematical models, we asked the participants to write down their rule of stacking or inserting blocks and compared that to the rule Darwin has deduced (Chapter 5).

2) *The IIBL methods of modeling a retrieval function with adaptive weights reduces the workload, i.e., reduces the number of demonstrations required to achieve the*

same amount of system performance, compared to k-nearest neighbor (k-NN) which assigns equal weights to all features:

3) *The privilege of interactive learning and a shared workspace is in that the system can achieve just-in-time learning. The teacher can interrupt and provide demonstration when it is actually needed while observing the learner's performance in real time based on the learned task model*:

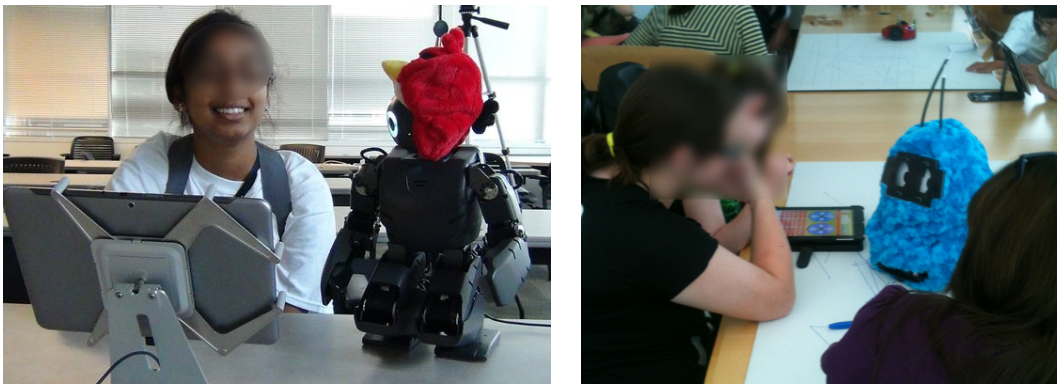
4) *The teacher's behavior changes based on the learner's model of the same task.*

4.6 Robotic Platform

In Figure 35, the two candidate robotic platforms are shown. In choosing the candidates, we carefully reviewed the following qualifications. First, the robots have to frequently travel to meet participants in their homes, clinics, institutions, or organizations, so portability was considered. Next, the robots need to share a workspace with a person, e.g., tablet computers, and therefore the size is limited. DARwIn-OP (Dynamic Anthropomorphic Robot with Intelligence-Open Platform) is 45cm(18 in) in height, weighs 2.9kg(6.4 lb), and has 3-DoF arms, 6-DoF legs, and a 2-DoF head with LEDs on its eyes and forehead [51]. Romibo is 32cm(12.6 in) in height, weighs 1.8kg (4 lb), has a mobile base, and a touch sensor across its body [110]. Both platforms are able to provide expressive vocal and gestural responses for social interaction. In this paper, we have shown results using DARwIn-OP as an evaluation platform. Henceforth, we will refer to DARwIn-OP as "Darwin" or "the robot".

Darwin is programmed with a range of verbal and gestural behaviors that are coupled with emotion indicators using the LED in its eyes. The behaviors are grouped into positive, negative, neutral, and idle states. During block stacking and inserting

experiments, Darwin generated a verbal script based on the retrieved solution (Chapter 5). When Darwin was engaged in tasks on a tablet, the verbal scripts were replaced with non-linguistic utterances (Chapter 6). The combinations of verbal and gestural behaviors are randomly generated within each behavior group. These groupings are based on prior studies that examined the effect of different behaviors on engagement [41]. The robot is also induced with a passive personality, meaning Darwin retracts its motions and waits for his turn whenever its human teacher reaches out to provide demonstrations or interact with the tablet.



(a) DARwIn-OP (b) Romibo

Figure 35: Robotic platforms

4.7 Summary

The motivation of the interactive instance-based approach to robot learning started from the necessity to incorporate features of the task that are diverse in their representations. As a means to gain knowledge about the given task's features, an interactive approach that borrows human intuition about the task was utilized. The keywords people use to convey task knowledge to others are the good representations of what the users would choose as task features. After the attributes of these features, i.e., the type of data, extraction methods, and distance metrics, are defined, instances

are created from demonstrations provided by the users. Such case-acquisition approach using interactive LfD is especially useful for modeling tasks that no prior knowledge was given to the system. Interactive methods also facilitate learning by closely monitoring the learner's progress and performance. In the process, the superior explanatory behavior of the instance-based framework informs the user with which instances generated the given prediction, which in turn provides a setting for the user to provide revised demonstrations to the learner at the right timing.

Designing an instance-retrieval mechanism is the most important issue in instance-based learning. Many forms of k-NN and case-indexing methods are used to measure similarities between instances, but since the retrieval mechanism requires significant amount of task-specific knowledge, generalizing a similarity model is hard to achieved. This dissertation research proposed to use weighted k-NN and suggested two methods to compute task adaptive weights. This distance measure is a weighted linear summation of each task-feature distance. The weights represent the relative contribution of the task features in making a prediction of the task behavior. The first approach used linear regression to find a vector of weights that maximizes a reward function. However, its incapacity to take into account nonlinear relationships between the dependent variables and each independent variable was revealed to be a major drawback in modeling tasks in which many of its features are dependent. Therefore, the second method relied on ANN's strength in modeling nonlinear behaviors. It was then necessary to work on methods like contribution or sensitivity analysis to add power to ANNs in their explanatory capacity. Here, three sensitivity analysis methods were explored with feedforward neural networks. Among the three methods, Shin's sensitivity measure and Lek's profile method showed an agreeing behavior in ordering the relative contribution of the features. Both methods measured the variation in the output variable's behavior in accordance to the change in each feature variable. Shin's method approached the problem by setting all connected weights of the input

variable in question to zero and measuring the deviation in the output. Lek's profile method computes the prediction of the output variables across the range of the feature variable of interest. All other inputs are assigned fixed values starting from their minimum, then successively at the 20-th percentile until maximum. The goal is to plot a set of input-output profiles across each input while other variables are held constant. During the deployment of the robot learner, the weight vectors produced from these two sensitivity analyses were averaged and used as toward designing a case-retrieval function.

CHAPTER V

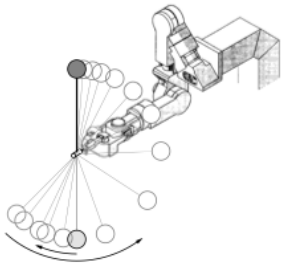
ROBOT LEARNER IN A SHARED PHYSICAL WORKSPACE

In this chapter, the proposed IIBL framework is applied to a play setting where partners agree upon a rule that arises as the play progresses. Interactive play with peers during childhood is the most primitive form of imitation and case-based learning. Through playing, individuals learn basic interaction skills, collaboration, patience and build understanding of others [94]. With respect to playing with others, a shared interest arises between playmates to make the play continuously entertaining, thus engaging the mind and creating opportunities for extended play over longer durations [45].

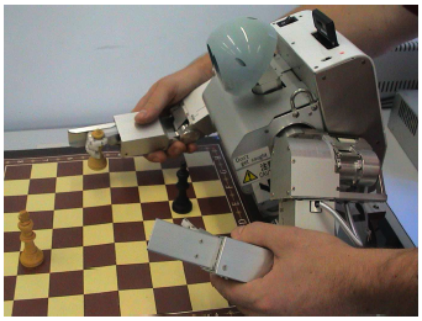
5.1 Introduction

Playmates share common objectives during play, whether it is stacking blocks or racing toy cars on a track. Shared play goals lead to shared play motions that are repeatedly observed until the end of play [25]. Developmental scientists, clinicians, and therapists have expressed interest in the idea of applying robotic platforms as an assistive therapeutic device, where robots are suitable for providing consistent and repetitive exposure to social interactions. For example, a child-like humanoid Kaspar has shown potential in encouraging autistic children to participate in an imitation play as a social mediator [99]. Physical-exercise robot coaches initiate a physical activity and observes the human execution [79]. The robot coach then assesses the participant's performance and provides feedback or adjusts the level of exercise intensity. These interactions occur in turns, and the robot engages the participant

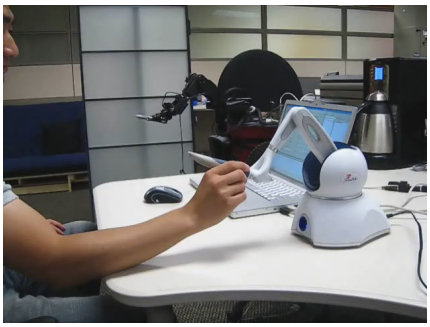
(a) Leonardo: learning gestures from the subject [21].



(b) Task-level learning demonstrates swing-up pendulum [6].



(c) Programming by demonstration [27].



(d) Transfer of human skills in learning hand writing [90].

Figure 36: Robot learning from demonstration.

in a social interaction for the duration of the exercise. Some features of the above robot systems, however, interfere with providing a long-lasting interaction relationship between human and robot. The robots are often teleoperated, and even with autonomous systems, the contexts of the tasks that the robot can provide are very limited. The goal of our research is to develop an automated learning system on a robotic agent that could observe, build knowledge of the task, and generate its own interaction behavior for a prolonged interaction.

From the object-play study conducted based on the first five levels of child's object manipulation [8], it was shown that the most prevalent forms of object-play with intended task objectives were inserting and stacking [92]. Learning to play with objects is closely aligned with robotics research focused on learning manipulation tasks from human demonstration (Figure 36). The researchers attempt to model the

human motions and create a mapping of the models to the physical embodiment of the robot. Brooks et al. [21] presented an interesting work in teaching interactive plays to a robot (Figure 36(a)). The authors used vocal- and gestural-social cues to teach the robot how to imitate human's facial expressions and motor movements. In the work of Atkeson and Schaal [6], it was mentioned that a direct imitation of human joints could become sensitive to noise. Therefore, the authors explored a task-level learning that used task models and reward functions to compute appropriate policies (Figure 36(b)). Calinon et al. [27] considered learning trajectories and constraints from demonstrations. The approach aimed at extracting relevant characteristics of the gestures that are needed to be reproduced. The motion data is encoded using a mixture of Gaussian/Bernoulli distributions that provide a measure of the spatio-temporal correlations across different modalities (Figure 36(c)).

In this chapter, the IIBL framework is applied to learning the play rule defined by the participants. Evaluation of IIBL performance was conducted using the video clips collected from various object-play scenes, and later an experiment was conducted with a robot learner that uses vocal and gestural primitives to communicate the generated solution.

5.2 Case-based Representation of Object Play

In order to select the features that contribute to describing or defining a task, a pilot study was conducted to examine what attributes of the block stacking and inserting tasks people chose to convey to others for collaboration. Six participants each watched six randomly selected clips of a thirty play-scene video set, and later were asked to explain the task to the experimenter with an emphasis on the elements they found were important in describing and differentiating the play tasks. An example play scene is depicted in Figure 37.

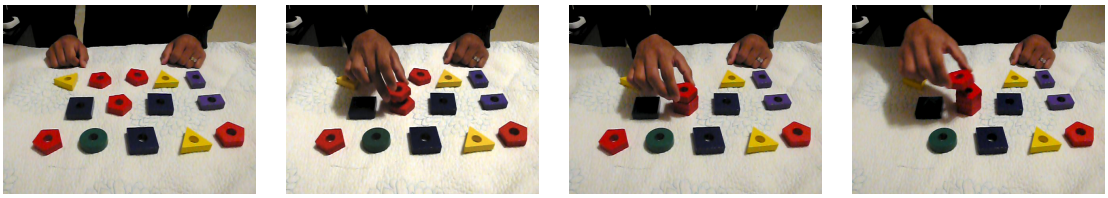
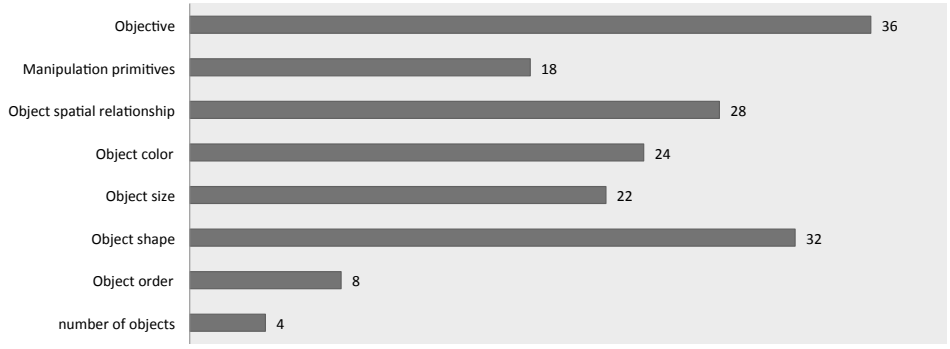


Figure 37: The participants were asked to execute a task with their intended task rules.

Statement	Objective	Manipulation primitives	Object spatial relationship	Object color	Object size	Object shape	Object order	number of objects
Put red blocks in the red bin and put blue blocks in the blue cup.	put(insert)		in	red, blue		bin, cup		
Stack all red blocks.	stack			red				all
Smaller cups are stacked on top of the larger ones.	stack		top		smaller, larger	cup		
Pick up the square blocks on the left and place them in the bin on the right.	place(insert)	pick up, left, right	left, right			square blocks, bin		
Align the opening of the box to the colored blocks and slide them in.	slide in (insert)	align, slide in	align, in	colored		box		
Cups are stacked in this order: orange, blue, and red.	stack			orange, blue, red		cup	orange,blue, red	3

(a)



(b)

Figure 38: (a) Extracting task features that appear as keywords in the participants' statements regarding the objective of the task. (b) Frequency of the task features that appear in participants' task-description statements (Sample size, $N_s = 36$).

In Figure 38, the most frequently observed features were extracted and categorized. The final list of task features used in the experiment are presented below. In

the following, the *primary-object* is an object being manipulated, and the *secondary-object* is a target object that the primary-object is stacked onto or inserted into.

Task Feature Descriptors

- *Task category*: The objective of the task. It is distinguished by an interaction between the primary- and the secondary-object.
- *Spatial object relationship*: The primary-object's continuously changing spatial information relative to the secondary-object is represented as a sequence of spatial-relationship primitives.
- *Discrete motion-primitive sequence*: The trajectory of primary-object manipulation is represented as a sequence of discrete motion primitives.
- *Primary- and secondary-object shape descriptor*: The shape descriptor (SD) is a vector that describes the scale- and intensity-invariant shape of an object.
- *Relative size*: The size ratio between the primary- and the secondary-object.
- *Primary- and secondary-object color*: The two dimensional hue-saturation color information of the object.

Detailed extraction methods, distance metrics, and adaptation methods of these task features are in the following section. Later, the spatial relationship of the primary and secondary objects was integrated within the sequence of manipulation motion primitives. Table 6 summarizes the loaded extraction and distance-metric modules for each task feature.

Table 6: The task-feature attributes (extraction methods, distance metrics, and adaptation methods) for the object-based play tasks as defined in the user-configuration file.

Problem Feature	Extraction Method	Distance Metric	Weight
Objective	HMM motion-primitive recognition	Binary	w_1
Shape descriptor	Split-region eigen value & eigen vector	Euclidean	w_2, w_3
Relative size	Color tracking & bounding region	Euclidean	w_4
Color	H-S color representation	Binary	w_5, w_6

Solution Feature	Extraction Method	Adaptation Method
Objective	HMM motion-primitive recognition	Binary
Manipulation-primitive sequence	HMM motion-primitive recognition and sequencing	Distance vector
Shape descriptor	Split-region eigen value & eigen vector	Euclidean
Relative size	Color tracking & bounding region	Euclidean
Color	H-S color representation	Binary

5.3 Feature-extraction Methods

The proposed CBR module is highlighted in Figure 39. In this figure, a new *problem* is introduced to the system. The problems, indicated by lightly shaded regions, describe what object operation has been executed in each turn. For instance, the new problem in the figure reads, an orange toy was inserted into a red toy, and the size ratio between the two toys is 0.0970. The vectors in the second and third columns define the shapes of the objects. The medium shaded regions define a *solution*. The solution is a set of objects chosen in the succeeding turn. Lastly, the dark shaded region is a *case ID*. The case ID, problem, and solution together define a *case*. The structure of a task case is illustrated under the case example Case-001 in Figure 39.

During the initial training session, cases are gathered to build a *task case-base*.

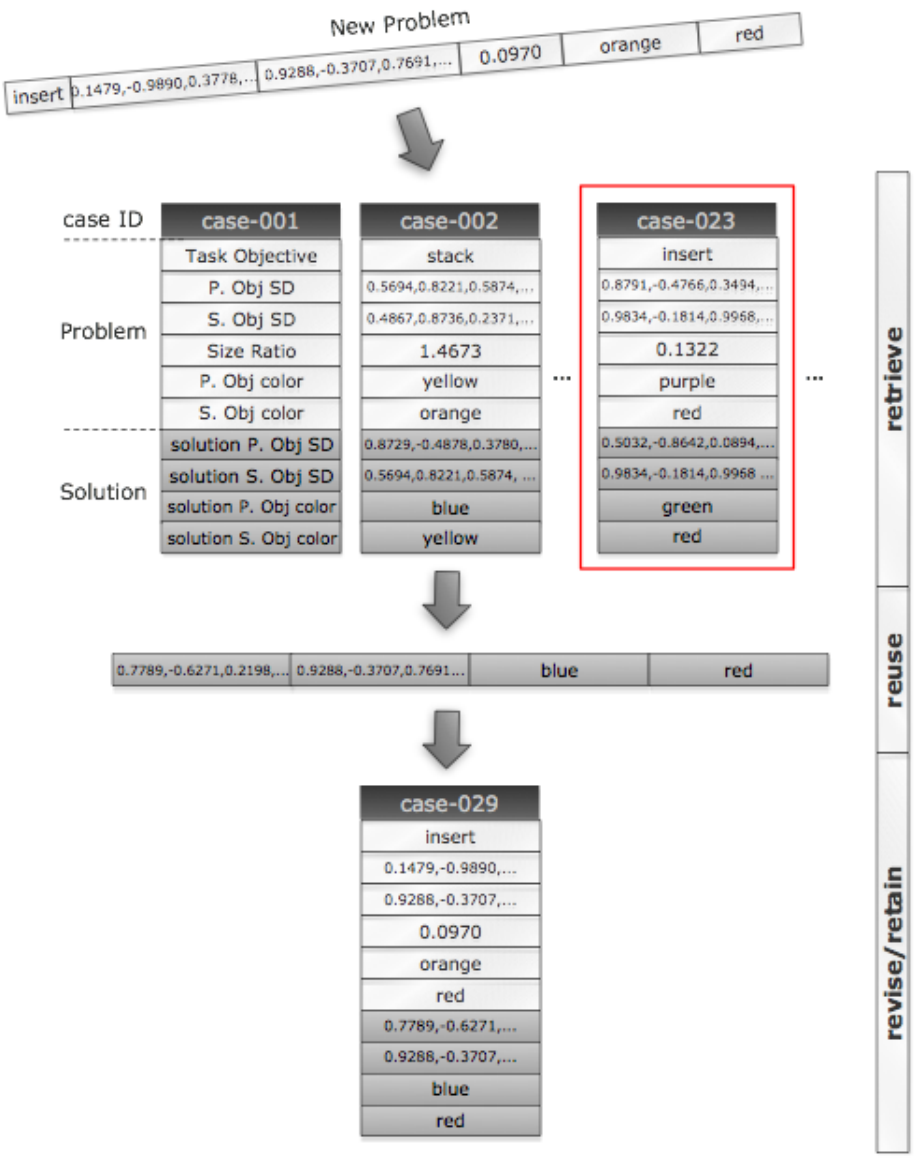


Figure 39: Illustration of the retrieve-reuse-revise-retain steps of the IIBL framework for the object-based play tasks. A new query is compared to the existing instances in memory, and the system seeks for a solution that will function as a starting point for generating a new prediction.

5.3.1 Shape Descriptor

Young children have an inborn ability to differentiate between shapes [66]. As children develop, shapes become a fundamental basis in building skills that will help them with

reading, writing, and mathematics. It is natural that many toys focus on training children with shape matching and recognition abilities.

Shape is one of the main attributes that characterize a task. To equip a robotic system with the ability to learn new tasks, an effective shape recognition algorithm should be developed. Play tasks that require shape matching, i.e, nesting cups or stacking rings, involves toys that vary in size and color, but share the same outlining shape. Popular object recognition techniques like Scale-Invariant Feature Transform [75] or Speeded Up Robust Features [11] extracts interesting features to provide a feature description of the objects. However, since many of the toys lack distinctive features because of their smooth surface and edges, these feature-based object description approaches are inappropriate for our work. Another issue is that, these algorithms are computationally expensive and hard to be implemented in real time. Template-based recognition algorithms such as the scale-invariant template matching approach [23, 63] may be executed faster, but it is still insufficient for real-time instance-based systems.

The goal of our shape matching process is to design a descriptor that is computationally inexpensive, has low-dimensional vector, and still has robust scale/intensity invariance. This enables the process for populating and finding the play case to occur in real time robustly. The shape descriptor we propose in this section computes dominant edge angles that describe the shape. It is like a rough sketch of the object with linear lines.

Edge detection refers to the process of identifying and locating sharp discontinuities in an image. The discontinuities are abrupt changes in pixel intensity which characterize boundaries of objects in a scene. The Canny edge algorithm [28] uses a gradient method which detects edges by looking for the maximum and minimum in the first derivative of the image. After the edge map is created using the Canny edge detector, the map is divided into 16 regions, and for each region, the dominant linear

edge angle is computed from the linear least squares fitting. The algorithm runs in six steps:

1. **Preprocessing**: The object region is segmented from the image, and blurred to remove noise.

2. **Finding gradients**: The edge is marked at the point where the gradients of the image have large magnitudes.

3. **Non-maximum suppression**: Only local maxima should be marked as edges.

4. **Double thresholding**: Potential edges are determined by thresholding.

5. **Edge tracking by hysteresis**: Final edges are determined by suppressing all edges that are not connected to a very certain (strong) edge.

6. **Finding Linearity**: The edge map of the object is divided into 4x4 regions. For each region, the dominant linear edge angle is computed by the linear least squares fitting algorithm.

First, each object region is cropped into sub-images from the scene after it is detected using the hue-saturation histogram back-projection in Section 3.3.3. The cropped image is then converted into a grayscale representation. Since edge detection is prone to noise, the raw image is smoothed using a filter. A Gaussian filter is used for this purpose with standard deviation $\sigma = 1.4$. The 2-D isotropic Gaussian has the form:

$$f(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}. \tag{33}$$

Since images are composed of discrete pixels, a discrete approximation of the above Gaussian filter is used to perform convolution with the image. Gaussian distribution is non-zero everywhere, which would require an infinitely large convolution kernel, but in practice it is effectively zero more than about three standard deviations from

the mean, and so we can truncate the kernel at this point. The discretized Gaussian filter $\mathbf{F'_d}$ with $\sigma = 1.4$ is as follows:

$$\mathbf{F'_d} = \begin{bmatrix} 0.0105 & 0.0227 & 0.0293 & 0.0227 & 0.0105 \\ 0.0227 & 0.0488 & 0.0629 & 0.0488 & 0.0227 \\ 0.0293 & 0.0629 & 0.0812 & 0.0629 & 0.0293 \\ 0.0227 & 0.0488 & 0.0629 & 0.0488 & 0.0227 \\ 0.0105 & 0.0227 & 0.0293 & 0.0227 & 0.0105 \end{bmatrix}, \tag{34}$$

where

$$\mathbf{F'_d}[i,j] = \frac{1}{2\pi\sigma^2} e^{-\frac{i^2+j^2}{2\sigma^2}}, \qquad i,j = -2\ , -1\ ,\ 0\ ,\ 1\ ,\ 2\ . \tag{35}$$

The discretized result of Gaussian does not sum up to 1 since only the partial information was used. Therefore, the above matrix is normalized so that the sum of all elements equals to 1. The resulting discrete 5x5 Gaussian filter is

$$\mathbf{F_d} = \begin{bmatrix} 0.0121 & 0.0261 & 0.0337 & 0.0261 & 0.0121 \\ 0.0261 & 0.0561 & 0.0724 & 0.0561 & 0.0261 \\ 0.0337 & 0.0724 & 0.0935 & 0.0724 & 0.0337 \\ 0.0261 & 0.0561 & 0.0724 & 0.0561 & 0.0261 \\ 0.0121 & 0.0261 & 0.0337 & 0.0261 & 0.0121 \end{bmatrix}. \tag{36}$$

The comparison between the continuous and the discrete approximation of the Gaussian filter is depicted in Figure 40.

Next, gradient magnitudes and directions are calculated at every single point in the image. The magnitude of the gradient at a point determines if it possibly lies on an edge or not. A high gradient magnitude means the intensities are changing rapidly, which implies an edge. The direction of the gradient shows how the edge is oriented. The standard Sobel operator is used to calculate the gradient. Here $\hat{\mathbf{I}}$ is

the object sub-image, and convolving the image with Sobel operators results in the gradients in the $x-$ and $y-$ directions, $\mathbf{G}_x$ and $\mathbf{G}_y$.

$$\mathbf{G}_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} * \hat{\mathbf{I}} \quad , \quad \mathbf{G}_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} * \hat{\mathbf{I}}. \tag{37}$$

The magnitude $|\mathbf{G}|$ and the angle θ of the gradient is computed by,

$$|\mathbf{G}| = \sqrt{{\mathbf{G}_x}^2 + {\mathbf{G}_y}^2}, \tag{38}$$

$$\theta = \arctan\left(\frac{\mathbf{G}_y}{\mathbf{G}_x}\right). \tag{39}$$

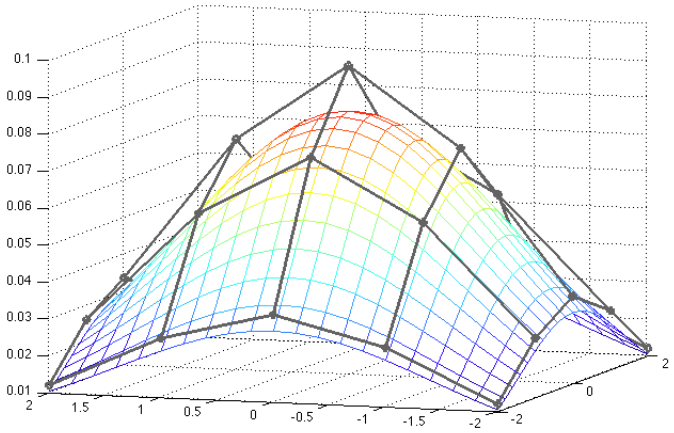


Figure 40: Discrete approximation (grey) of the continuous Gaussian filter (color) ($\sigma = 1.4$).

The first step in non-maximum suppression, which the purpose is to convert the blurred edges in the image of the gradient magnitudes to sharp edges, is to get an

rough idea of the edge direction by rounding the gradient angle θ to the nearest 45°. Now, the gradient in each pixel will point towards 90°, 45°, 135°, or 0°. The non-maximum suppression, like the name implies, is basically done by preserving all local maxima in the gradient image and deleting everything else. The gradient magnitude of the current pixel is compared to the edge strength of the pixels in the positive and negative gradient direction. For example. if the gradient direction is 135° (NW), the pixel is compared with the pixels to the northwest and southeast. If the gradient magnitude of the current pixel is the largest, preserve the edge. If not, suppress (i.e. remove) the value. This will give a thin line in the output image.

Finally, double thresholding is used as a means of eliminating streaking. Streaking is the breaking up of an edge contour caused by the operator output fluctuating above and below the threshold. If a single threshold, T1 is applied to an image, and an edge has an average strength equal to T1, then due to noise, there will be instances where the edge dips below the threshold. Equally it will also extend above the threshold making an edge look like a dashed line. To avoid this, the Canny edge algorithm uses two thresholds, a high and a low. Any pixel in the image that has a value greater than T1 is presumed to be an edge pixel, and is marked as such immediately. Then, hysteresis step determines any pixels that are connected to this strong edge pixel and that have a value greater than T2, which are also selected as edge pixels.

The final edge image is divided into N regions. For each region, the dominant edge angle is computed, and these N angles, which forms the shape descriptor vector, characterize the brief shape of the object. Since the object images are subdivided into equal number of regions, the size of the object image, or the size of the object itself has little influence on this shape descriptor. Hence, the descriptor is size invariant, but not invariant to rotations, which sufficiently serves the purpose of the research in question. The method of computing the dominant edge angle in each region is to find the best fitting linear line $y = \beta_1 x + \beta_2$ of the data, i.e., the edge pixels. The solution

to this problem can be rephrased as finding a linear line that the sum of distances (errors) between the data and the line is minimized. If the region has n number of edge pixels with coordinates $(x_1, y_1), \cdots, (x_n, y_n)$, we have an overdetermined linear system:

$$\begin{aligned} \beta_1 x_1 + \beta_2 &= y_1 \\ \beta_1 x_2 + \beta_2 &= y_2 \\ &\vdots \\ \beta_1 x_n + \beta_2 &= y_n, \end{aligned} \tag{40}$$

which can be written in a matrix form:

$$\begin{bmatrix} x_1 & 1 \\ x_2 & 1 \\ \vdots & \vdots \\ x_n & 1 \end{bmatrix} \begin{bmatrix} \beta_1 \\ \beta_2 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}, \quad \text{or} \quad \mathbf{X}\boldsymbol{\beta} = \mathbf{y}. \tag{41}$$

The best solution $\hat{\boldsymbol{\beta}}$ is the one that minimizes

$$\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} E(\boldsymbol{\beta})^2, \tag{42}$$

where

$$E(\boldsymbol{\beta})^2 = \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 = (y_1 - \beta_1 x_1 - \beta_2)^2 + \cdots + (y_n - \beta_1 x_n - \beta_2)^2. \tag{43}$$

The above function $E(\boldsymbol{\beta})^2$ is the minimum at the point where $\dfrac{\partial E(\boldsymbol{\beta})^2}{\partial \beta_1}$ and $\dfrac{\partial E(\boldsymbol{\beta})^2}{\partial \beta_2}$ equals to zero.

$$\begin{aligned} \frac{\partial E(\boldsymbol{\beta})^2}{\partial \beta_1} &= -2x_1(y_1 - \beta_1 x_1 - \beta_2) - \cdots - 2x_n(y_n - \beta_1 x_n - \beta_2) &= 0, \\ \frac{\partial E(\boldsymbol{\beta})^2}{\partial \beta_2} &= -2(y_1 - \beta_1 x_1 - \beta_2) - \cdots - 2(y_n - \beta_1 x_n - \beta_2) &= 0. \end{aligned} \tag{44}$$

The above equation can be written in matrix form as

$$(\mathbf{X^T X})\boldsymbol{\beta} = \mathbf{X^T y}, \quad \text{or} \quad \begin{bmatrix} \sum_{i=1}^n x_i^2 & \sum_{i=1}^n x_i \\ \sum_{i=1}^n x_i & n \end{bmatrix} \begin{bmatrix} \beta_1 \\ \beta_2 \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^n x_i y_i \\ \sum_{i=1}^n y_i \end{bmatrix}. \tag{45}$$

Once $\hat{\boldsymbol{\beta}}$ is solved, the dominant angle for the region is calculated using

$$\phi = \arctan \hat{\beta}_1, \tag{46}$$

which forms the shape descriptor vector $\mathbf{p}$,

$$\mathbf{\Phi} = [\phi_1, \phi_2, \cdots, \phi_N]^T. \tag{47}$$

The overall process of creating an edge map and computing a shape descriptor is shown in Figure 41. Here, the image is divided into $N = 16$ regions.

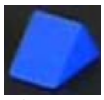
(a) original image (b) edge image (c) shape descriptor

Figure 41: The edge map of the object image is computed using the Canny edge detector and divided into sub-regions. For each region, the dominant edge angle is calculated using the linear least squares fitting.

5.3.2 Shape-descriptor Evaluation

To evaluate the proposed shape descriptor, we have collected 40 cropped images that consist of 32 object images and 8 random images from the scenes. The 32 object images fall into one of the object groups in Table 7. Different rings are grouped together since they share a common shape, and the same rule applies to the stacking cups. All other objects are individually grouped.

Table 7: Objects used for shape descriptor evaluation

object1 (rings):				
object2 (cups):				
object3~8:	 object3	 object4	 object5	 object6
	 object7	 object8		

Each image in the test set is compared to all other test images including itself. The graph in Figure 42 shows the distances between the sample object and some other object images in the test set. Recall that the range of the normalized distance metric for the shape descriptor is $0 \leq \delta(\mathbf{p_i}, \mathbf{p_j}) \leq 1$. As shown in the figure, the size variation of the object does not affect the shape matching: The distance is 0.0202 when the object is reduced to 50% and 0.0176 when enlarged to 150%. If the distance between the shape descriptors is within a threshold, the two objects can be seen as a match. According to the experiment, setting the distance threshold to $\lambda = 0.22$ resulted in the best performance.

Table 8 is a confusion matrix for the recognition result of the whole test set with $\lambda = 0.22$. Shaded regions indicate which object each image belongs to, and the numbers inside the parenthesis show how many object images there are in the test set. The recognition rate was computed by multiplying the probability of the recognized true positives and the probability of the recognized true negatives. The performance of the proposed shape descriptor was 0.84, i.e, 84%.

$$\text{SD recognition rate} = \frac{\sum(\text{recognized true positives})}{\sum(\text{actual true positives})} \cdot \frac{\sum(\text{recognized true negatives})}{\sum(\text{actual true negatives})}. \tag{48}$$

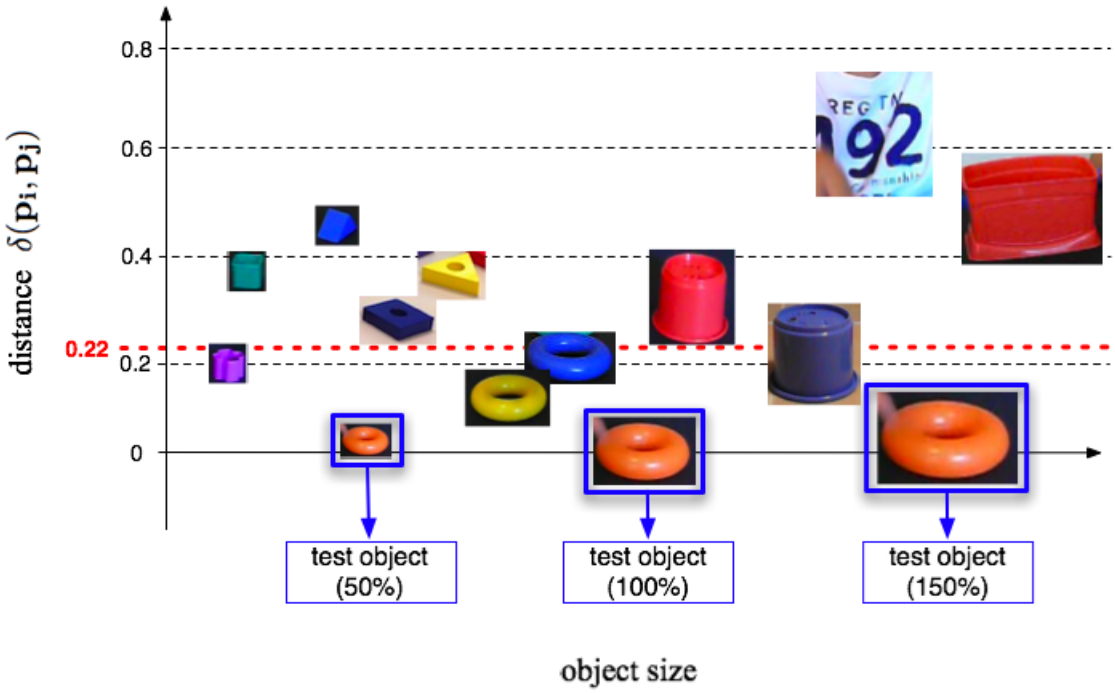


Figure 42: Test object in Figure 41 is being compare to other objects. Experiment result shows that the shape descriptor is size-invariant, and the best distance threshold is $\lambda = 0.22$.

Table 8: Confusion matrix for the shape matching result using the shape descriptor. Shaded regions indicate which object each image belongs to, and the numbers inside the parenthesis show how many object images there are in the test set. ($\lambda = 0.22$)

	object1 (4)	object2 (4)	object3 (4)	object4 (4)	object5 (4)	object6 (4)	object7 (4)	object8 (4)	misc (8)	# of false neg.	# of false pos.
image1	4	2			1	2				0	5
image2	4	2				2				0	4
image3	4	1		1	1	1				0	4
image4	4	1	2		1	1				0	5
image5	1	4		1	1	1				0	4
image6	2	4				2				0	4
image7	2	4		1	1					0	4
image8	1	4				2				0	3
image9			3	1						1	1
image10			3		1					1	1
image11	1		4		2					0	3
image12	1		4	2	1					0	4
image13			1	4				1		0	2
image14	1		1	4		1			1	0	4
image15		2		4	1	1		1		0	5
image16			1	4	2		1			0	4
image17	1		1		4				1	0	3
image18	2	2	1	1	4	1		1		0	8
image19			1	1	4	2				0	4
image20			1	1	4	1	1			0	4
image21	3	2			1	4				0	6
image22				1	1	3				1	2
image23	1	1		1	2	3				1	5
image24	2	2				4				0	4
image25							3			1	0
image26				1	1		3			1	2
image27							2			2	0
image28							4			0	0
image29								4		0	0
image30				1	1			3		1	2
image31				1				4		0	1
image32								3		1	0
image33-40				1	1						

5.3.3 Task Objective

Exploratory play emerges by four months of age and includes behaviors such as simple repetitive object manipulations [29]. One of the later evolving types of play include relational play such as putting blocks in a cup. Such relational play defines our task-objective feature, which is the main objective of the current play task. Whether it is a stacking or an inserting operation, it describes how a person manipulates the primary-object in relationship to the secondary-object.

As described in Chapter 3, the task-objective feature is recognized using the sequence of the most likely hidden Markov models (HMM) of the play primitives.

5.3.4 Size and Color Descriptor

The size ratio between the primary-object and the secondary-object and the colors of the toys are also computed during preprocessing. The object colors extracted during a preprocessing step for object tracking, which was explained in details in Section 3.3.3. The color is represented in two dimensional hue and saturation value, and is put into the closest color bin. Thus, the color γ is expressed as

$$\gamma = \{\text{color0, color1, color2, color3, color4, color5}\}. \tag{49}$$

which typically represents red, blue, green, yellow, purple, and orange in the play case. When detecting color, the size of the toy is approximated using the number of pixels, and the size ratio ρ is computed as

$$\rho = \begin{cases} \dfrac{N_{A_i}}{N_{B_j}}, & \text{if } N_{A_i} \leq N_{B_i} \\ -\dfrac{N_{B_i}}{N_{A_j}}, & \text{if } N_{A_i} > N_{B_i} \end{cases}. \tag{50}$$

where N_{A_i} is the average number of pixels calculated over 15 frames that represent the primary-object with color i, and N_{B_j} is the average number of pixels that represent

the secondary-object with color j. Note that the cases with $N_{A_i} = 0$ or $N_{B_i} = 0$ indicate no objects were detected.

5.4 Evaluation of IIBL

5.4.1 Experimental Setup

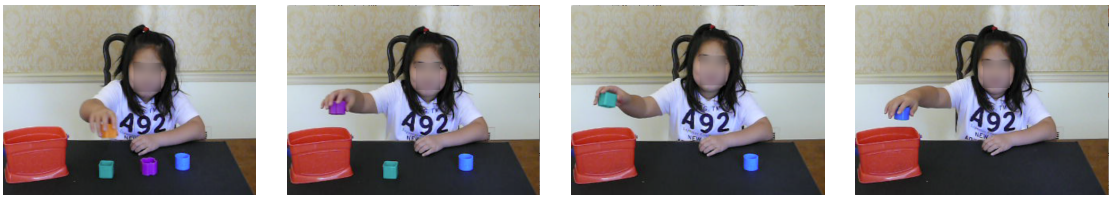
First, we collected 30 video sessions of inserting and stacking tasks for evaluation. The videos were recorded from various locations including participants' homes, clinics, and our lab. The participants' ages ranged from 5 to 40 (sample size $N_s = 30$, mean $m = 15.17$, standard deviation $\sigma = 9.07$), and each session consisted of 5 to 12 turns of object manipulations. The participants were given a set of varying toy objects and were asked to generate a task rule prior to engaging in a session. Some examples of task rules were inserting all blocks of certain size into a bin, stacking blocks with a color sequence, and inserting blocks to same colored cups. After the first task execution, the toy objects were rearranged and the participants were asked to repeat the task following the same rule. The first task sessions were used for training, and the latter sessions were collected for evaluation.

5.4.2 Task Case-base Training

The training phase is a period dedicated to collecting cases to create an initial task case base. During this period, the teacher demonstration is used to seed the task policy exploration process modeled by the retrieval-mapping function. The case base could be trained prior to

A video session used for training an inserting task is illustrated in Figure 43. In this example, the participant is inserting blocks one at a time in the red bin. In Table 9, each turn is segmented into case-descriptor features, which consists of a task objective, manipulation trajectory, shape descriptors and colors of the objects (presented as figures for better illustration), and a size ratio of the two. Each turn becomes the *problem*, while the succeeding turn becomes the *solution*. The problem

and the solution descriptor pair forms a *task case* and is stored in the *case base* (Figure 44).



(a) (b) (c) (d)

Figure 43: A training video session: A participant demonstrating an inserting task.

Table 9: Corresponding cases of the training session illustrated in Figure 37.

Subject#:02 Session#:03 Total-time:15 sec #-of-turns:4

Turn #	Task Objective	Primary-Obj	Secondary-Obj	Size Ratio
1	insert			0.0958
2	insert			0.0879
3	insert			0.0930
4	insert			0.0737

5.4.3 Embodiment Mapping

The derived solution is converted into vocal script and gestures to generate interaction behaviors on a robotic agent. Here, participants are engaged in a similar task as shown in the video sessions. After the participant initiates a turn, the robot responds by turning its head and pointing towards the next primary and secondary objects while speaking a script generated using adapted solution features (Figure 45).

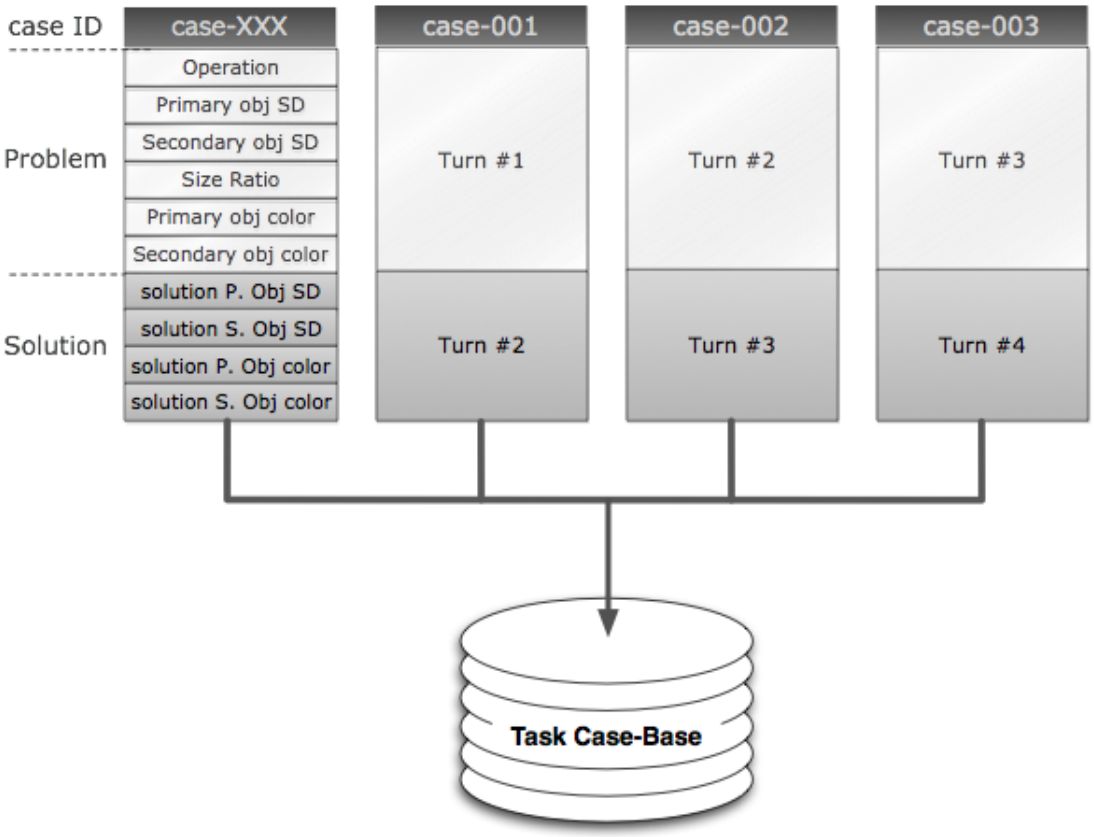
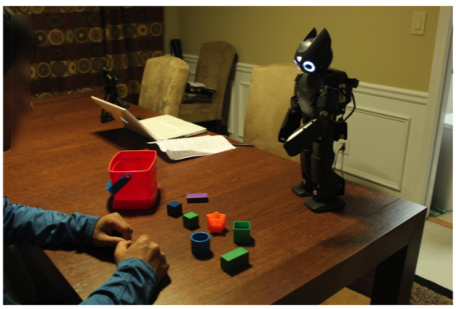


Figure 44: Each turn and its subsequent turn in Table 9 become a *problem* and a *solution* to form a *task case*, which is stored in the *task case base*.



(a) Vocal Script: "Next, I would pick up the *green object* and move *up-left-down* then *insert* to the *larger red object*."

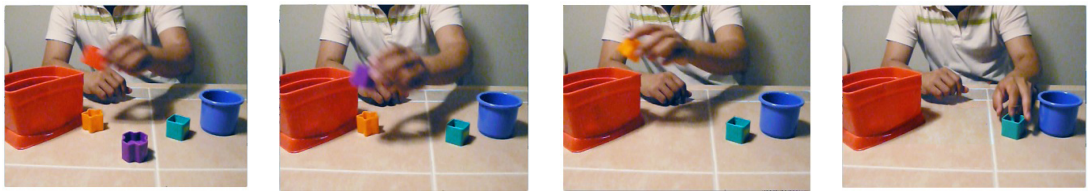
(b) Vocal Script: "Next, I would pick up the *blue object* and move *upright-right-down* then *stack* on the *larger orange object*."

Figure 45: The embodiment mapping generates multimodal interaction including vocal and gestural behavior on a robotic agent.

5.5 Evaluation Results and Discussions

In this section, the evaluation results of the learning performance of the IIBL system are presented. The solutions generated using the two instance-retrieval methods, k-nearest neighbors (k-NN) and the proposed IIBL feature weighting, were compared.

The instances were acquired during participants' demonstrations of the task. The participant's provided demonstrations according to the task goal they had specified before the trials. These case bases were evaluated with the video sessions collected for testing. The snapshots of a task scene in Figure 46 was one of the test sequences used for evaluations with the IIBL system built for the example training task in Figure 43. The problem descriptor for each individual turn is listed in Table 10. This specific session shared the same task objective with the example task, but the objects presented were slightly different. Among the five objects used during the training phase, four objects re-appeared and one was removed from this scene. In addition, two objects were introduced to the task space. With this setup, we were able to evaluate the following conditions:



(a) (b) (c) (d)

Figure 46: Participant demonstrating an inserting task used for evaluation of the IIBL framework.

1. When there exist an exact match to the problem and the retrieved solution. (Figure 46(b))

2. When there exist an exact match to the problem, but no exact match to the retrieved solution. (Figure 46(c) & Figure 46(d))

Table 10: Query instances of the testing session illustrated in Figure 46.

Subject#:05 Session#:01 Total-time:7 sec #-of-turns:4

Turn #	Task Objective	Primary-Obj	Secondary-Obj	Size Ratio
(a)	insert			0.0813
(b)	insert			0.1399
(c)	insert			0.0895
(d)	insert			0.1100

3. When there is no exact match to the problem. (Figure 46(a))

In the following, a qualitative comparison of the solutions generated using k-NN and IIBL is presented.

5.5.1 Using k-NN Equally Distributed Feature Weights

1. *When there exist an exact match to the problem and the retrieved solution.*
   This condition corresponds to Case (b) in Table 10. The introduced problem is depicted below:

Problem	insert			0.1399

The following problem and solution pair was retrieved using k-NN:

Retrieved Problem	insert			0.0879
Retrieved Solution	insert			0.0930

The primary and secondary objects that match the retrieved solution were present in the current scene. The following object pair was successfully recognized:

Adapted Solution	insert			0.1100

2. *When there exist an exact match to the problem, but no exact match to the retrieved solution.*

   This condition corresponds to Case (c) and Case (d). First, the introduced problem in Case (c) was:

Problem	insert			0.0895

Once again, the closest case was successfully retrieved.

Retrieved Problem	insert			0.0958
Retrieved Solution	insert			0.0879

However, now in the scene in Figure 46(c) there was no longer an exact primary-object match to the retrieved solution. Still, the reuse step attempted to find

the best match among the objects that were present. The predicted solution was:

Adapted Solution	insert			0.3303

The prediction was not suitable for the task objective, and the reason the system selected this object was because the distance between the shape descriptors was relevantly smaller than the other remaining objects. Now, let's take a look at the introduced problem and the retrieved case in Case (d).

Problem	insert			0.1100

Retrieved Problem	insert			0.0930
Retrieved Solution	insert			0.0737

The objects in the retrieved solution did not exist in the scene (Figure 46(d)), so the closest solution was generated:

Adapted Solution	insert			0.3303

Again, though the shape matching was not one of the task objectives, the equally distributed weights had no control over prioritizing one feature over another.

3. *When there is no exact match to the problem.*

   This condition corresponds to Case (a). The problem introduced was:

Problem	insert			0.0813

   This time, a case that directly matched the introduced problem did not exist in the case base. The closest case that was retrieved was:

Retrieved Problem	insert			0.0958
Retrieved Solution	insert			0.0879

   The retrieved case aligned with the task goal, and the closest solution was generated:

Adapted Solution	insert			0.1399

   Since the new problem-solution pair adhered to the task objective, this new case was stored in the case base during the revise/retain step of the IIBL.

5.5.2 Using IIBL Feature Weighting

The objective of the given task examples, Figure 37 and Figure 46, was to put small blocks into the big red bin. The trained weight values are depicted in Figure 47. Note that the shape and the color of the secondary object received highest weights along with the task objective and the object-size ratio.

The retrieved cases and the adapted solutions for each condition are as follows:

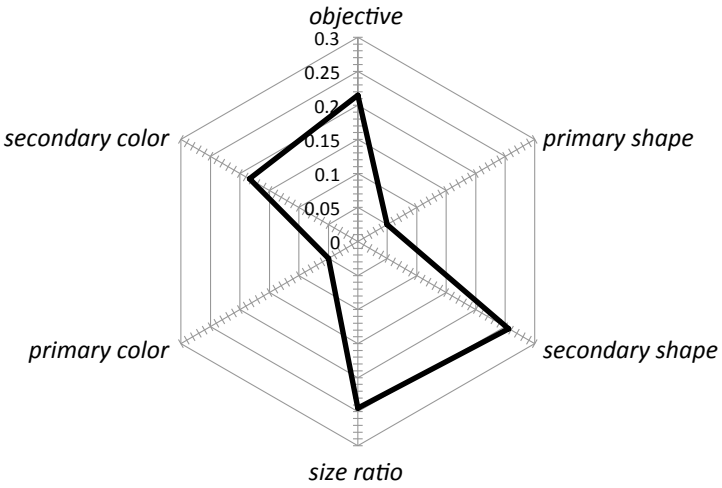


Figure 47: IIBL weights trained after nine demonstrations of an inserting task.

1. *When there exist an exact match to the problem and the retrieved solution.*

Problem	insert			0.1399

Retrieved Problem	insert			0.0958
Retrieved Solution	insert			0.0879

The retrieved instance according to the new distance metric predicted the following solution:

Adapted Solution	insert			0.0895

Which was not an exact match to the retrieved solution, but the newly generated solution conforms to the task objective no less.

2. *When there exist an exact match to the problem, but no exact match to the retrieved solution.*

Problem	insert			0.0895

Recall that when the weights were equally distributed, the system generated an outlier solution for this problem. This time, the retrieved case and the adapted solution were:

Retrieved Problem	insert			0.0879
Retrieved Solution	insert			0.0930
Adapted Solution	insert			0.1100

Similarly, when Task-(d) was queried to the system, an instance that had the most similar size ratio was retrieved:

Problem	insert			0.1100

Retrieved Problem	insert			0.0958
Retrieved Solution	insert			0.0879

However, since the only object left in the scene (Figure 46(d)) was not close enough to the retrieved solution, the system returned "no match".

3. *When there is no exact match to the problem.*

   Though the primary-object in this problem hadn't been seen before in the system, the trained feature weights retrieved a previous instance and its solution that most closely mimic the size ratio of the two objects.

Problem	insert			0.0813

Retrieved Problem	insert			0.0879
Retrieved Solution	insert			0.0930

Adapted Solution	insert			0.0895

Once again, the predicted solution was within the task objective.

5.5.3 Discussion

In the previous sections, we overviewed how the solutions generated from k-NN and IIBL differ in an inserting scenario. IIBL showed better tolerance and adaptation

to generating a correct prediction when the solution space of the task domain is incomplete. Figure 48 compares various play scenarios executed by the participant and the task behavior deduced by the IIBL framework. For each task, IIBL extracted problem-solution pairs from the participants' demonstrations and performed generalization through training a set of feature weights according to their contribution toward categorization. Figure 49 shows the result of generalization in the form of trained feature weights using multivariate linear regression for various tasks intended by the participants.

Next, error rates were analyzed for the three methods that were used to provide task-behavior predictions. The direct computation method used what was observed as a query to find the best matching primary and secondary objects remaining in the scene without retrieving any solution. The k-NN used equally weighted sum of feature distances as a retrieval mechanism, and lastly, IIBL trained the feature weights during user demonstration which was used towards designing a retrieval function. In Figure 50, the task scenarios intended by the participants were categorized into 6 different general inserting and stacking task objectives. The numbers inside the parenthesis indicate the numbers of different scenarios in that category. For the video evaluations, the solutions generated by each system were manually labeled right and wrong. During human-robot interaction, participants were asked to perform the solution generated by the robot if the robot's prediction was right, and to demonstrate the correct behavior if not. The methods that used prior solutions as a starting point for making predictions about a new solution performed significantly better than a direct computation, and IIBL produced less errors than k-NN.

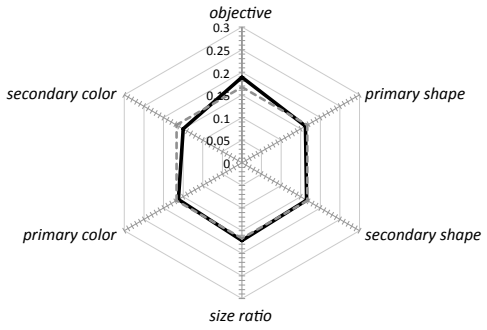
5.6 Summary

In this chapter, the proposed IIBL framework was used to model object-based play scenarios. Various block inserting and stacking tasks were the target scenarios as

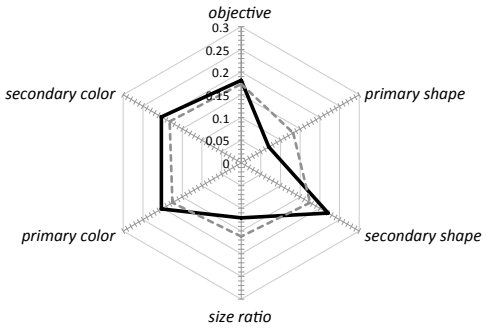
(a) (b) (c)

	Play (a)	Play (b)	Play (c)
Problem (Play Scene)	Subject *stacks blue cup* on *orange cup*. Size ratio 0.85	Subject *inserts blue block* into *red bin*. Size ratio 0.11	Subject *inserts orange block* into *red bin*. Size ratio 0.14
Retrieve			
ttCBR retrieved Problem	stack blue cup on orange cup	insert blue block into red bin	insert blue block into red bin
ttCBR retrieved Solution	stack red cup on blue cup	insert purple block into red bin	insert purple block into red bin
Reuse			
ttCBR adapted Solution	stack {red cup} on blue cup	insert {purple block} into red bin	insert {purple block} into red bin

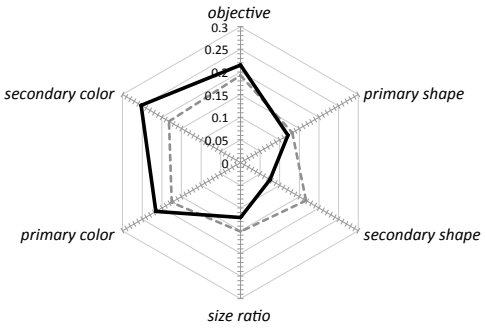
Figure 48: Various block inserting and stacking tasks modeled using an IIBL framework.



(a) "Stack cups from the largest to the smallest."



(b) "Put red objects in red bin, and blue objects in blue cup."



(c) "Stack all green blocks."

Figure 49: Task objectives generated by the participants and the results of trained adaptive feature weights. The dashed lines indicate the results after using 50% of the training data, and the solid lines show the results after using 100% data.

these scenarios involve the most prevalent forms of object manipulation in the first five levels of child's development. Furthermore, such shared play environment with block objects encouraged participants to create their own task objectives which in turn provided a great setting for IIBL framework evaluations.

The process of task modeling involved conducting studies to extract motion primitives and designing instance-based framework to incorporate other features of the block stacking and inserting tasks. A study was conducted to survey what components of the task people would use to convey task knowledge to others. The participants were introduced to the current skills and perceptual capabilities of the robot,

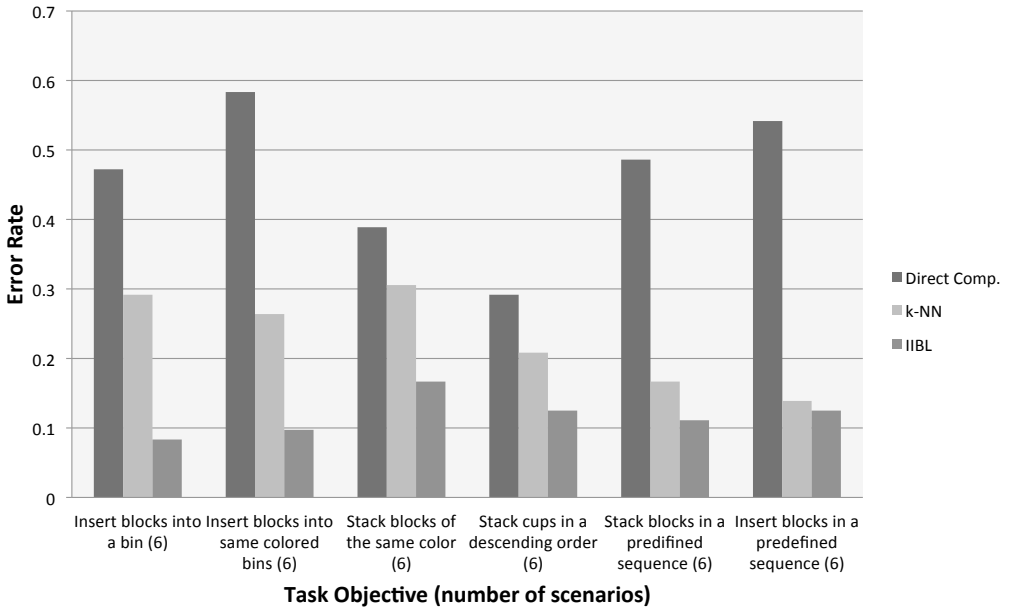


Figure 50: Comparison of error rates using direct computation, k-NN, and IIBL for generating task behaviors in various task scenarios. The scenarios were categorized into general inserting and stacking task objectives.

and their feedback triggered the development of additional feature-extraction methods. The task-primitives research presented in Chapter 3 was used to extract task objectives and manipulation behaviors, and the object-tracking algorithm was used to obtain object-related attributes such as the color and size. A fast scale-invariant shape descriptor was developed to compute a vector of regional edge angles to represent the shape of an object.

The IIBL framework was evaluated against various play scenarios. The participants created their own task goal and provided demonstrations of the task. A retrieval function was designed for every task objectives by computationally modeling the degree of contribution of each instance feature. The task features were assigned with weights that maximized the overall reward function. Afterwards, the generated solutions of the IIBL framework was compared to the predicted solutions using the k-NN.

A qualitative approach evaluated the system in three categories of possible query conditions, and the results showed that the IIBL method provided better adaptation to the task goals. A quantitative result reported that the instance-based methods, i.e., IIBL and k-NN, took less time to produce a solution compared to a direct computation method which used no prior information to generate predictions. IIBL also outperformed both k-NN and direct computation in producing solutions that conform to the task goal intended by the participants. A robot learner equipped with the IIBL framework successfully engaged in tasks with participants by suggesting the next moves using vocal and gestural behaviors.

Though the IIBL framework successfully produced better and faster predictions of the task turns compared to other methods, the current setup in a shared physical workspace posed difficult challenges to fully engage robot learners with children. Since our primary goal regarding the application domain is to deploy robot learners in various settings including children's homes, play clinics, and classrooms, the necessity of additional sensors placed around the workspace and the need to calibrate them defeated our purpose. Also an unpredictable behavior of children was difficult to take into account when designing the system. Tasks were often not conducted within the robot's field of view, and the system not performing robustly affected the quality of the interaction. To address these limitations, a new experimental system design was proposed using a commercially available touchscreen tablets.

CHAPTER VI

ROBOT LEARNER IN A SHARED TABLET WORKSPACE

One of the key elements for building a long-term robotic companion is incorporating the ability for a robot to continuously learn and engage in new tasks. Utilizing a defined workspace that provides various shared content between human and robot could assist in this learning process. In the previous chapter, a robot learner was able to learn and adapt to different scenarios of object-based play through an interactive instance-based learning (IIBL) method. In this chapter, we explore integrating a touchscreen tablet and a robot learner for engaging the user during human-robot interaction scenarios. We discuss results from the *Angry Darwin Expedition*, in which our robot Darwin learned to play a strategic game "Angry Birds" from various users. During a six-month period, over 130 people interacted with the robot learner including 90 children, among which 33 participated in the formal experiment.

In addition to measuring the learning performance of our robot learner, we measure how the varied system's learning models leverage the level of participant's engagement. The robot learner's domain-independent core reasoner follows the structure of instance-based learning which addresses the issues of acquiring knowledge, encoding cases, and deducing a retrieval metric. The system utilizes demonstrations provided by the user to auto-populate the knowledge base through natural interaction methods and encodes instances based on the feature structure provided by the human teacher. We have proposed two models for designing an adaptive-weighting retrieval function, first using linear regression in the feature-distance space and second through sensitivity analysis that is measured by hiding each feature node in the system's neural

networks.

6.1 Introduction

In this chapter, the user teaches a task to the robot in a shared tablet workspace and intuitively monitors the robot's behavior and progress in real time through a tablet environment. In this setting, the user is able to interrupt the robot and provide necessary demonstrations at the moment learning is taking place, thus providing a means to continuously engage both the participant and the robot in the learning cycle. First, the challenges we faced while the robot was engaged in a physical shared workspace are discussed, and the benefits of restraining the task space using a tablet and utilizing its virtual space for real-world applications are introduced.

6.1.1 Challenges

In the real world, human-robot interaction doesn't necessarily happen in face-to-face, one-on-one situations, nor does it take place in a controlled environment where robot sensors collect and interpret data as expected. In fact, when we deployed our robot in Chapter 5, we faced the following challenges.

- Unpredictable environment: In addition to the main environmental perception using RGB data from the camera, we explored methods of combining RGB-D data (RGB and depth information from infrared depth imaging device, Kinect [34]) for improved object tracking and auditory extraction (speech and voice-direction recognition from a microphone) for detecting start and an end of a demonstration. Though the system performed well in a lab setting, when the robot system was brought to the clinics and various home environments, we faced unpredicted challenges that prevented the system from performing robustly. Often the space was limited to place the sensors in optimal locations to oversee the task scene, some objects obstructed the sensor's view, or the room was poorly lit.

- Unpredictable behavior: The unpredictable behavior of children and their caregivers affected the robustness of the system. The tasks were often conducted outside the system's field of view. In sessions with children with cognitive or physical disabilities, a clinician or a caregiver would typically sit close to the participant. The presence of this additional person often affected the results of visual tracking and the outputs of processed auditory data. System not performing robustly affected the quality of the interaction that the participants experienced.

Using a now commonly available touchscreen tablet as a shared workspace, these challenges were overcome or reduced. When conducting and sharing tasks on the tablet, the user and the robot exchange interaction while minimizing the uncontrollable variables mentioned above. We have placed our robot Darwin in open-house events, exhibitions, play-therapy centers, and at homes, in which the system performed robustly while learning and improving from demonstrations provided by the participants. The advantages of a tablet workspace are discussed in the following section along with reviews of previous literature using tablets as research platforms.

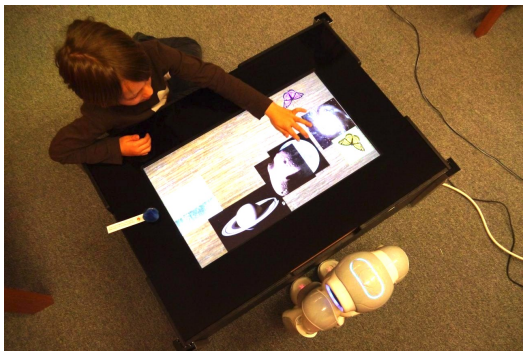
6.1.2 Tablet Shared Workspace

It is reported that robots can enhance user experience through functioning in conjunction with applications (apps) on smart devices. Popchilla [14] (Figure 51(a)) combines an interactive drawing application with a robot that generates motion and sound responses to user's input on the tablet. The robotic music-listening companion [55] (Figure 51(b)) produces on-beat motions to the music playing from the smartphone. The robot's rhythmic behavior makes the person feel like they are sharing the experience, and the person perceives the event as more enjoyable. Other examples include robots to which users can mount their smartphones that engages the users by animating their devices [13] (Figure 51(c)).

(a) Popchilla: an interactive robot with a drawing app [14].

(b) Travis: a robotic music-listening companion [55].

(c) Romo: a smartphone mobile robot [13].



(d) Sandtray: providing context to unstructured social human-robot interaction [10].

Figure 51: Robot companions enhance user's experience with smart devices.

As a research platform, touchscreen devices function as shared workspaces and reduce perceptual uncertainties. In LfD, there are several methods for recording a teacher's demonstration [4, 20, 85]. In these scenarios, the uncertainty of environmental perception always poses a difficult challenge, including tracking the teacher's motions or recognizing the human's social cues. The tablet platform reduces such uncertainty since the touchscreen provides quantified sensor data from the gestural behavior of the user. The development environment and the vast amount of apps on the market facilitate the process of designing and implementing a task with controllable modalities. Such benefits of deploying a touchscreen-based medium for studying interactions between human and robot have also been discussed in [10] (Figure 51(d)).

In their work, the touchscreen setup provides context to unstructured social human-robot interaction. The research is focused on emerging social behaviors of the child while the embodiment of the robot facilitates naturalistic interaction on the part of the child.

This dissertation research attempts to solve the limitations that the previous research efforts haven't addressed. Until now, most tablet-based robots exhibited simple reactive behaviors, were teleoperated, or were limited to conducting a single task. We focus on increasing the utilization of a single robot platform by introducing a robot learner that can maintain knowledge of multiple tasks. Based on our previous work, which addresses the efficacy of coupling tablets as a shared workspace with a robot learner for HRI studies [93], we present a system that could be easily configured to engage robots with tablet apps.

6.2 Task-Feature Representations

In IIBL, demonstrations are recorded as instances which are formulated as problem-solution pairs. In a tablet workspace, the structure of a solution consists of start and end locations of touch-gesture sequences. The easiest way to extract feature information from the tablet is to design the app so that it sends data packets at some sampling rate. Deciding what data to extract from the task depends on the discretion of the user. In the following, we describe in detail the tablet task that is used throughout the evaluations. Two pilot studies were conducted to observe what group features the users thought were was sufficient to learn the task, and how a retrieval function is trained using the features the pilot testers chose.

6.2.1 Task Description: *Angry Darwin Expedition*

We have applied the proposed IIBL framework during *Angry Darwin Expedition*, in which our robot, Darwin, learned to play a strategic game "Angry Birds" on a shared tablet workspace from various users. During a six-month period, over 130

Figure 52: The IIBL framework was applied to a strategic game on the tablet.

people interacted with our robot learner including over 90 children, among which 33 participated in the formal experiment.

The task's objective was to shoot the bird to destroy all enemies either by directly aiming at them or knocking down the surrounding structures to collapse them (Figure 52). Though the task may seem simple, it is actually a quite complex task if we were to mathematically model the physics behind the game components, such as the bird, enemies, wooden planks and stone barriers. However, for humans with natural ability to process spatial information, this becomes a task a young child could solve with trial and error. Thus, though a precise policy cannot be generated, time and computational cost for modeling a task is greatly reduced by reusing the solutions from previously observed instances.

6.2.2 Pilot Study I: Task-feature survey

A pilot study was conducted with eight participants to select task features that they would want the robot learner to extract instances from. The participants each played

four different game scenarios as shown in Figure 52. The goal of the task was to not just complete each level, but to maximize the score. Figure 53 shows the features the participants listed after conducting the task.

	Bird launch orientation & intensity	Score	Level	# of (remaining) lives	# of (remaining) enemies	Enemy locations	Wooden beam & plank locations	Stone beam & plank locations	# of structures	Whether the demo led to a successful completion
Participant 1	o	o	o	o						o
Participant 2	o				o	o	o		o	o
Participant 3	o	o	o	o	o	o				
Participant 4	o	o		o	o	o				
Participant 5	o	o			o	o	o	o		
Participant 6	o	o	o	o	o	o	o	o		
Participant 7	o	o	o		o	o	o			
Participant 8	o	o	o	o	o	o	o	o		
Total Votes	8	7	5	5	7	7	5	3	1	2

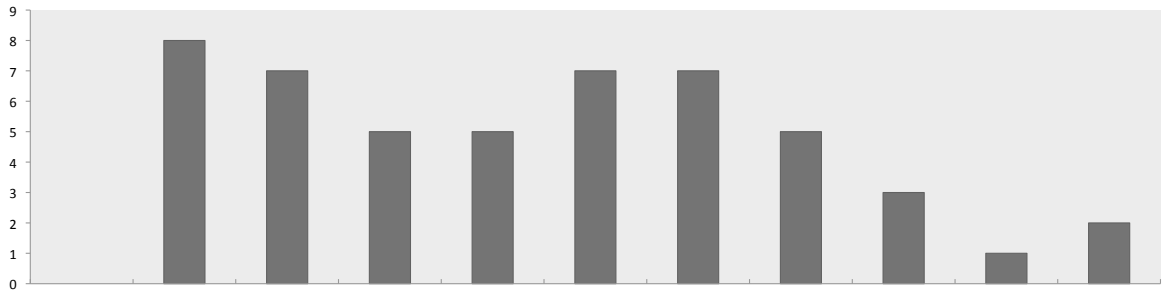


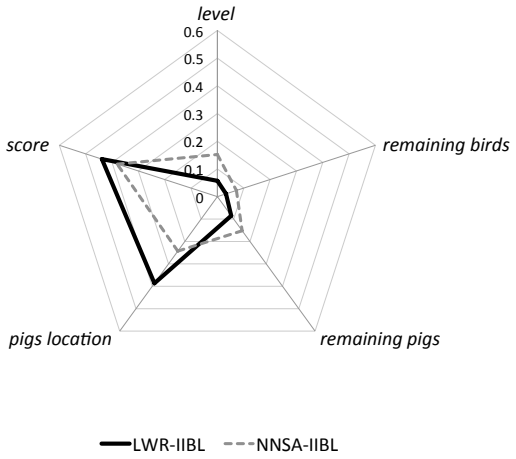
Figure 53: Task features listed by pilot-study participants.

Among the listed features, the bird's launching angle and intensity are obviously the solution features of this task. The score, number of remaining pigs, and their locations received the most votes among the problem features. Note that the feature sets of the participants #3 and #5 are equal in size and include the three most voted features. At the same time, the two sets include different group of features among the next most voted features: the level, remaining number of birds, and the location of the structures. In the following, another pilot study was conducted for training retrieval functions using these two feature sets.

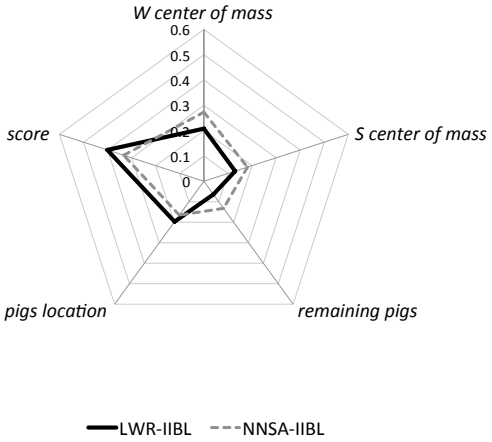
6.2.3 Pilot Study II: Training retrieval functions

The training data was gathered from eight pilot-study participants. The participants conducted four subtasks each, and the weights were trained using the two IIBL

methods: locally weight regression (LWR-IIBL) and neural network-based sensitivity analysis (NNSA-IIBL). Figure 54 shows the result using two different feature sets derived from the earlier study. In all methods, the score is the most dominant feature. This aligns with the fact that the success of the task is driven by the highest game score, which has high correlation to how many enemies were destroyed. In Section 6.4, we will use the four retrieval functions derived from the pilot study to conduct evaluations of the IIBL framework.



(a) Feature set I (Participant #3)



(b) Feature set II (Participant #5)

Figure 54: Two feature sets suggested by the participants, and the trained weights using IIBL methods.

6.3 Robot Design

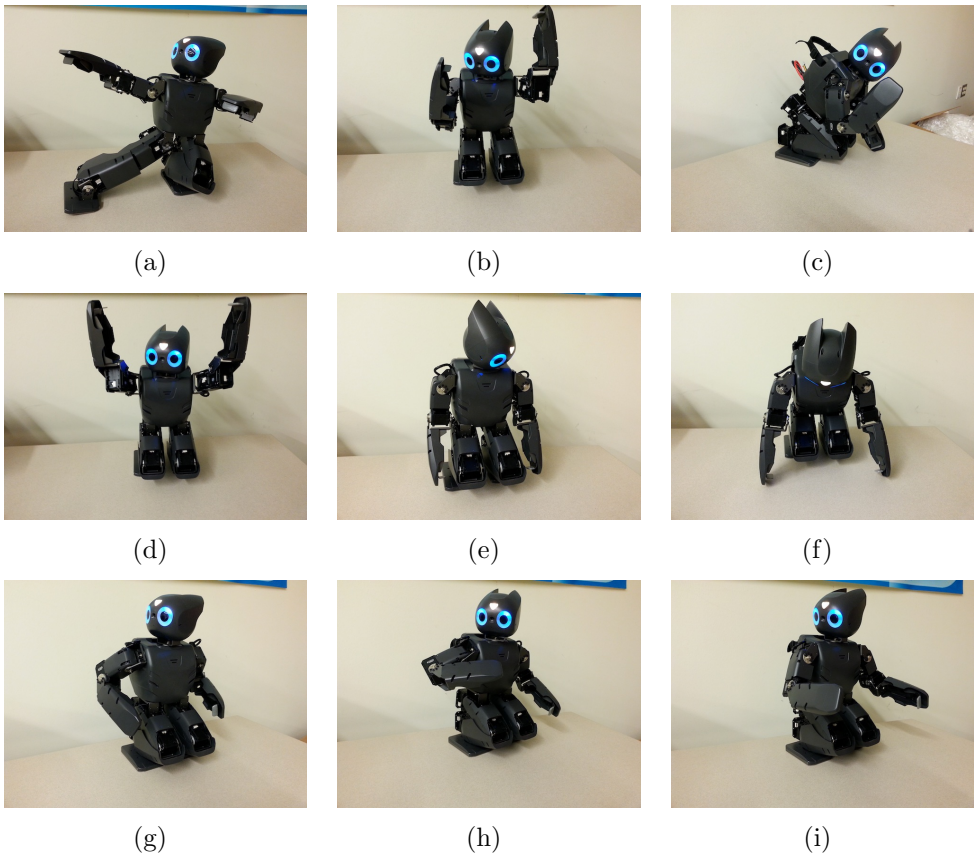
Once again, Darwin was chosen as a robotic platform to conduct experiments with participants on a shared tablet workspace. Darwin performed very robustly throughout the *Angry Darwin Expedition* in various settings, successfully delivering its learning behavior. First, Darwin's manipulators were programmed to manipulate objects on the screen. Darwin's physical restraints made it difficult for the robot to provide consistent touch pressure across the entire region of a tablet surface from one sitting position, and addressing the issue was outside the scope of this dissertation. Instead, Darwin was designed to practice synthesized touch behavior through wireless communication to the tablet. Along with the manipulation behavior, Darwin's gaze was directed towards the object movement on the tablet. Figure 55 shows the robot manipulating an object on the screen.

Figure 55: Darwin demonstrating hand-eye coordination while manipulating a red object on the tablet workspace through synthesized touch events. The gaze following helps the user feel the robot is more attentive and interactive.

Additionally, Darwin's vocal and gestural behaviors were designed to express four different states: positive and negative emotions, conversation mode, and idle. Darwin's vocal behavior mimics that of R2-D2's [1]. Each behavior group holds six to eight different vocal and gestural primitives. Some of the gestural behaviors are depicted in Figure 56. When retrieved, a random combination of vocal and gestural primitives

[1] R2-D2 is a robot character in the Star Wars universe. R2-D2's mechanical voice expresses emotion through intonation in its beeping sound effect.

is formed within the group. The robot is induced with a passive personality, meaning Darwin will retract its motions and wait for his turn whenever the teacher reaches out to provide demonstration or interact with the tablet.



(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

Figure 56: Darwin's gestural behaviors: (a)-(d) Positive emotion, (e)-(f) Negative emotion, and (g)-(i) Conversation mode

6.4 IIBL Performance Evaluation

For evaluation, we recruited 33 participants (mean age m=18.27, standard deviation σ=8.56) including 19 children (m=12.26, σ=4.16). The participants were to teach a virtual game, shown in Figure 52, to Darwin. We analyzed data collected during various events on campus during a two-month period. Groups of local school students and younger children, some with special needs, were invited to observe and participate in various experiments conducted in our research group.

Figure 57: Participants interacting with the robot learner. The experiment was conducted in an open-house styled setting with a group of local school children. Each participant engaged in two experiments in which the robot demonstrated different case-retrieval strategies.

For this experiment, participants were asked to teach two robots a strategic game on the tablet (Section 6.2.1), in which the player had to control the launching angle and the power of a bird to destroy enemies either by directly aiming at them or

knocking down the structures. The structure of the game makes various strategies possible to complete each level within a given number of attempts. A total of 33 sets of instance bases, including a total of 1,596 demonstrations and an average of 48.36 demonstrations per participant, were collected for evaluations. For training, datasets collected during the pilot studies in Section 6.2.2 and Section 6.2.3 were used.

In this section, the performances of the IIBL methods are evaluated against k-NN in their effectiveness and efficiency. Twelve problem scenarios in Figure 58 were used in the evaluation using 33 sets of case bases collected from the participants.

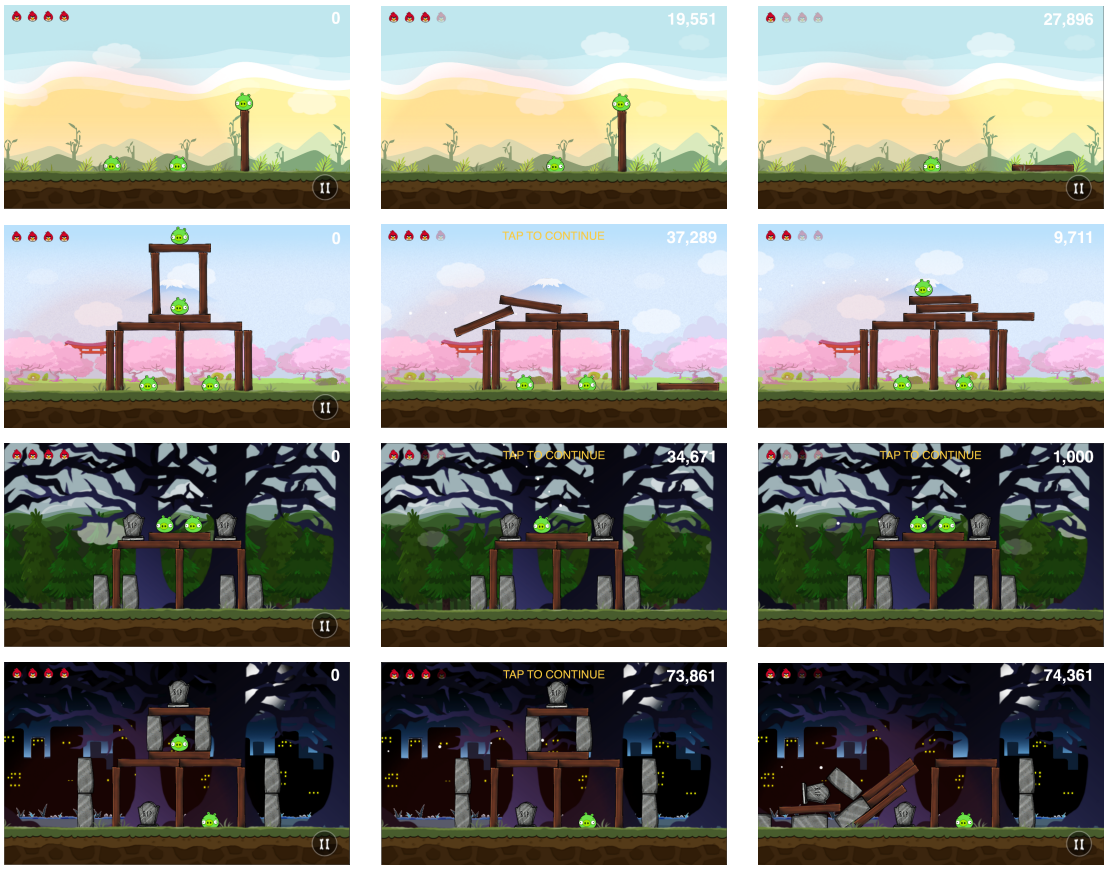


Figure 58: Twelve problem scenarios of the task used for evaluations.

6.4.1 Evaluation I: Effective Learning

The first hypothesis in regards to the learning performance was that the IIBL methods produce comparable task performance against the average performance of the demonstrator. In Table 11, performances of generated solutions from k-NN, LWR-IIBL, and NNSA-IIBL methods are compared to an average performance of the demonstrations provided by the participants. With varying k (number of retrieved cases), distances are computed between the query point and the problems in the case base using each retrieval method. Then the performance of each retrieved solution is evaluated using a logarithm of the earned game score.

Table 11: Mean performance (log(score)) of instance-retrieval methods compared to participant demonstrations with the feature set I.

k	k-NN	LWR-IIBL	NNSA-IIBL	Participants
1	4.14±2.23	5.12±0.93	5.49±0.61	3.75±2.02
2	4.02±2.02	4.97±0.76	5.15±0.43	-
3	4.13±1.72	4.78±1.08	4.91±0.21	-
4	3.96±1.46	4.89±0.86	4.76±0.09	-
5	3.11±1.87	4.12±0.82	4.34±0.12	-
6	2.79±0.92	3.82±0.44	3.86±0.13	-

Overall, the result shows that the average performance gradually decayed as k increased, and the confidence interval was larger when the solution depended on less retrieved instances. IIBL methods outperformed k-NN and produced more stable results (smaller confidence interval). When $k = 4$, IIBL performed on average 21.84% better than k-NN and 28.67% better than the average performance of the participants' demonstrations. The performance of the participants was averaged over all demonstrations given throughout the experiment which included the ones that were not necessarily successful or high quality. A rather large confidence interval of the

teacher's performance reflects such fact.

Table 12 shows the evaluation result using a different feature set. The previous observations are consistent with this feature as well. At $k = 4$, IIBL performed 11.87% better than k-NN and 15.90% better than the participants' demonstrations on average.

Table 12: Mean performance (log(score)) of instance-retrieval methods compared to participant demonstrations with the feature set II.

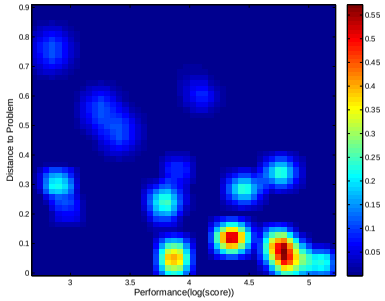
k	k-NN	LWR-IIBL	NNSA-IIBL	Participants
1	4.26±1.92	4.72±1.21	5.04±0.83	3.05±1.94
2	3.98±2.13	4.26±0.86	4.67±0.42	-
3	3.72±1.74	3.92±0.86	4.15±0.27	-
4	3.16±1.32	3.32±0.72	3.75±0.32	-
5	2.67±1.21	3.25±0.43	3.25±0.21	-
6	2.42±0.86	3.06±0.32	3.19±0.15	-

When distances between the query point and the instances in the case base are plotted, it clearly shows which cases are likely to be retrieved. In Figure 59, the regions of instances with the highest probability of being retrieved are more red than others, gradually transitioning into blue regions that are less likely to be selected. While k-NN in both feature sets shows scattered plots of likely retrieved instances, IIBL methods show better consistency between the likely retrieved solutions and their resulting performance when applied to a given problem. In other words, the instances retrieved by the IIBL methods have higher probability of producing the best performance, while k-NN doesn't guarantee that the instances most similar to the query point will generate good performance. The instances retrieved by k-NN differed from the ones retrieved by the IIBL methods, while the cases selected by the two IIBL methods were much similar. The two methods also agreed well on the

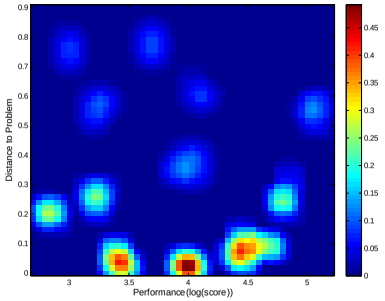
ordering of the relevance of the input variables. According to the plots, NNSA-IIBL produced more stable results (standard deviation σ_d from the linear fitting line = 1.15) compared to LWR-IIBL ($\sigma_d = 5.67$) among different query points. The evaluation results suggest that NNSA-IIBL is more stable and performs slightly better than LWR-IIBL.

6.4.2 Evaluation II: Efficient Learning

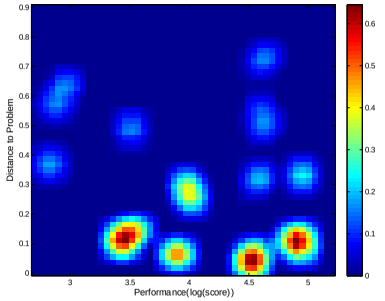
Next, the second hypothesis regarding the learning efficiency has been evaluated: We hypothesized that the IIBL methods reduce workload, i.e., reduce the number of demonstrations required to achieve the same amount of system performance, compared to the k-NN approach. We gradually increased the size of the case base with the instances collected from the participants and measured each method's performance. A total of 1,596 instances were added four at a time in a random order. Each method's performance was measured against the twelve problem scenarios in Figure 58. The result with $k = 4$ is depicted in Figure 60. According to the experiment, IIBL methods' performance increased faster then k-NN. LWR-IIBL took 162 instances, NNSA-IIBL took 153 instances, and k-NN took 212 instances to reach within the 95% convergence for all query points. On average, IIBL required 34.60% less instances to solve all twelve problems with the best performance. If a sufficient number of cases populate the problem space, IIBL and k-NN's performance will eventually converge. That is, if the number of high-performance demonstrations for each possible problem increases, both methods would start retrieving instances with similarly high-yielding solutions with little effect of the feature set. However, exploring all possible problems will increase the teacher's workload significantly. Therefore, the IIBL algorithm effectively increased the learning performance while reducing the teacher's workload in teaching a task to the robot.



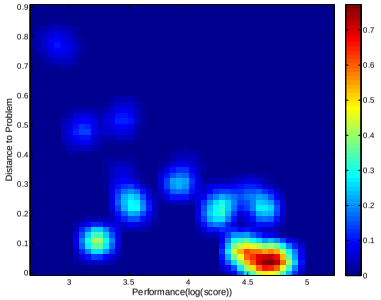
(a) k-NN, Query I, Feature set I



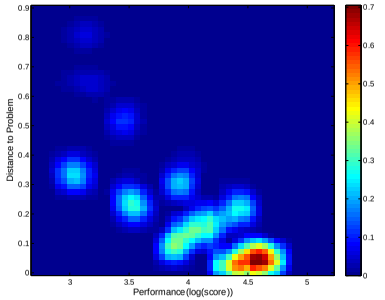
(b) k-NN, Query I, Feature set II



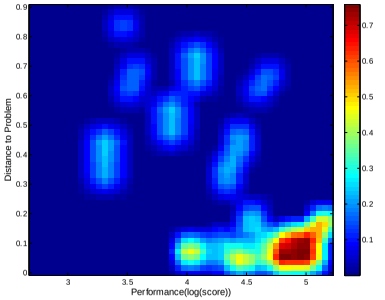
(c) k-NN, Query II, Feature set I



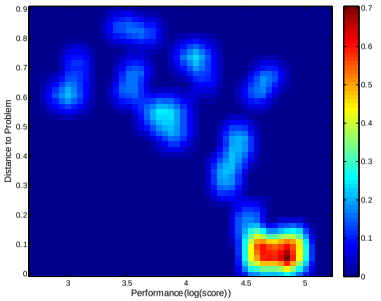
(d) LWR-IIBL, Query I, Feature set I



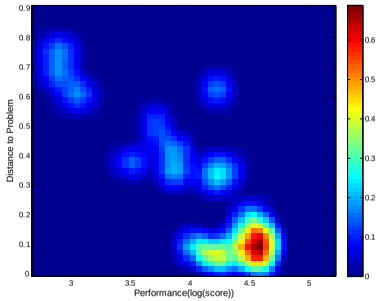
(e) LWR-IIBL, Query I, Feature set II



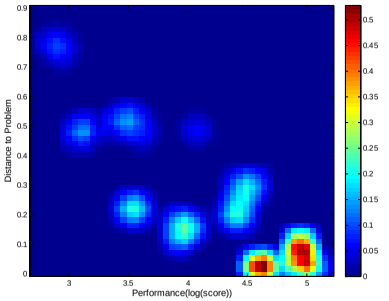
(f) LWR-IIBL, Query II, Feature set I



(g) NNSA-IIBL, Query I, Feature set I



(h) NNSA-IIBL, Query I, Feature set II



(i) NNSA-IIBL, Query II, Feature set I

Figure 59: Retrieval probability of instances depicted along the distance from given query points I and II, and their corresponding performances when the retrieved solutions are applied to each problem. Two different feature sets were used that were suggested by the participants. Both IIBL methods successfully retrieved instances that are best performing, while k-NN method's retrieved cases are scattered across the performance range.

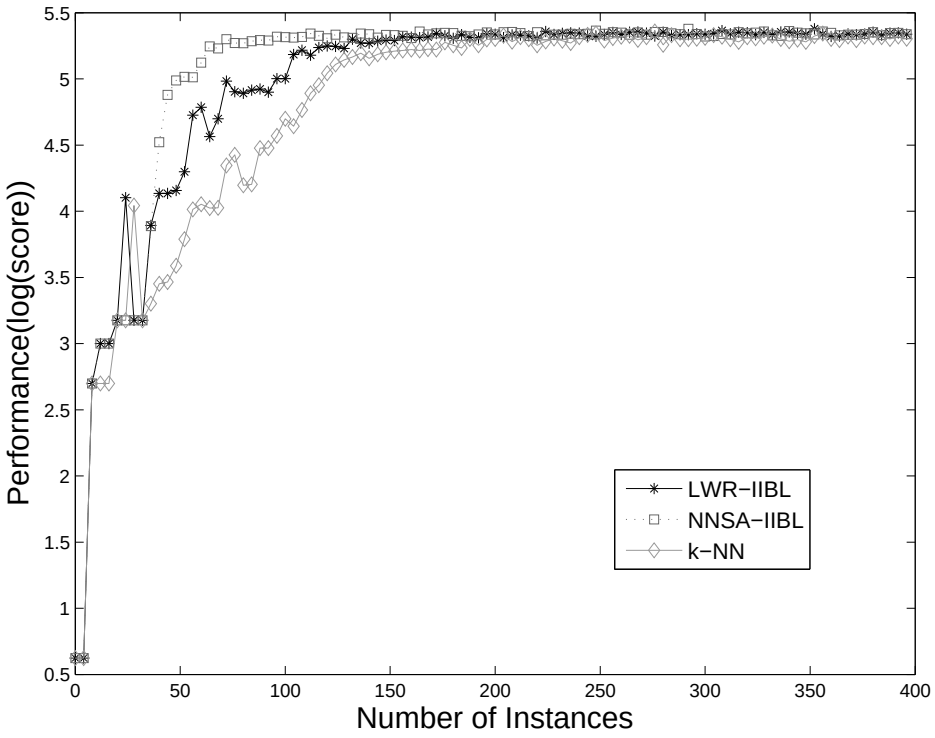


Figure 60: Evaluation of the case-base size and the performance of IIBL methods compared to k-NN. NNSA-IIBL learned the fastest while k-NN generally required more instances to produce similar performance.

6.4.3 Discusssion

In Chapter 4, we discussed the possible trade-offs between the linear regression and the sensitivity analysis with neural networks approach to IIBL. While LWR assumes a linear relationship between the variables, ANN is known for its capacity to model nonlinear behaviors. Both methods generated successful predictions for task-feature contributions, and the retrieval functions recommended by the IIBL methods produced superior outcomes than k-NN. Throughout the application presented in this chapter, ANN-based approach to IIBL showed more stable performance compared to LWR. However, ANN generally requires more data points to achieve good performance.

In this section, we have evaluated IIBL methods by comparing them to the

teacher's demonstration and k-NN. The two hypotheses regarding the effectiveness and efficiency of the IIBL methods were validated, and we have observed that IIBL produces higher performance with less instances than traditional instance-based learning using k-NN. The two IIBL methods, LWR-IIBL and NNSA-IIBL, agreed in the ordering of the two most relevant features and the least contributing feature (Table 13).

Table 13: IIBL methods's ordering of the features evaluated by their contribution to the task.

Feature Set I	LWR-IIBL	NNSA-IIBL
score	1	1
level	4	3
remaining birds	5	5
remaining pigs	3	4
pigs location	2	2

Feature Set II	LWR-IIBL	NNSA-IIBL
score	1	1
wooden-structure center of mass	2	2
stone-structure center of mass	4	3
remaining pigs	5	5
pigs location	3	4

The average performance of NNSA-IIBL was not significantly better than LWR-IIBL, but it was observed that NNSA-IIBL exhibited much more stable outputs. The decision of which IIBL methods to use depends on the nature of the given task. If an online training needs to be performed during learning, LWR-IIBL might be a

better choice in terms of training time. If the task requires better consistency in the performance, NNSA-IIBL would be a better approach. Also, if the task variables are dependent or possess unknown correlation, NNSA-IIBL would model the system better.

6.5 IIBL Human-robot Interaction Studies

Prior to a formal study, over 130 people interacted with our robot learner during a six-month period including over 90 children. These people participated in one of our "Angry Darwin Expedition" events in which we placed Darwin in various exhibitions, open-house events, and workshops. Attendees were encouraged to interact with Darwin while teaching him to play Angry Birds. A sign-up sheet was used to collected interested attendees' contact information which was later used to recruit experiment participants. From this list, we successfully recruited 33 participants (mean age m=18.27, standard deviation σ=8.56) including 19 children (m=12.26, σ=4.16). Children's gender demographics were: child-female ($n = 7$, $m = 12.71$, $\sigma = 3.86$) and child-male ($n = 12$, $m = 11.08$, $\sigma = 4.56$).

Participants were asked to participate in two sessions involving interaction with a tablet application. Touch-based gestures with the tablet were logged, and two video cameras were placed to record the sessions. The log and videos were later used for system evaluation. In the first session, Session I, participants were asked to interact with the game, without the robot learner, while the experimenter was present. The goal of Session I was to collect baseline data for evaluating differences when interacting with a person versus with a robot. In Session II, the participants were asked to interact with the robot by teaching the robot learner how to play the game. The participants were asked to repeat the teaching activity with two different robot learners in Session II. The experimental protocol is listed in Appendix A. Robots were prepared in three settings: Robot A (IIBL), Robot B (k-NN), and Robot C (random retrieval

from the current case base). To prevent ordering effects, participant groups and the robot-learner combinations were counterbalanced as in Table 14.

Table 14: The number of experiments conducted in each combination and ordering of the two robots.

First Robot	Second Robot	# of participants
B	A	10
C	A	7
A	B	10
A	C	6

In both sessions, it was the participants' first time interacting with the experimenter and the robot. The structure of the Angry Darwin game makes various strategies possible to complete each level within a given number of attempts. The instructions given by the experimenter to the participants was strictly scripted to avoid any influence it might cause to the participant's experience. The script was as follows:

> *Now, I'd like you to teach Darwin to play the same game. Just teach him in the same manner you would teach your friend. Provide Darwin with demonstrations on how to solve each level. Whenever you reach out to provide a demonstration to Darwin, he will wait for his turn. Continue teaching each level until you are satisfied that Darwin has learned the level well enough, or you think Darwin has stopped learning. Later, I want you to show me what you have taught Darwin, and collaboratively solve each level with him. Darwin may or may not try to communicate with you, and he may not use human language. Afterwards, I will ask you some questions about your experience teaching a task to Darwin.*

First, we evaluate how the participant's teaching behavior adapts to the robot's different learning strategies. Next, to address the objective of determining if there is

a difference in the emergence of behaviors initiated by an individual when conducting a task with a person or with a robot, social behaviors initiated by the participants were measured as the length of time when eye contact was made or when verbal- or gestural- interaction behaviors were observed. Lastly, to determine whether the length of social behaviors differs between typically-developing children and children with autism spectrum disorder (ASD), we conducted a pilot study with two children diagnosed with ASD.

6.5.1 Evaluation III: Teaching-behavior Adaptation

The third hypothesis was that the participant's teaching behavior changes based on the robot learner's learning method and performance. First, the efficiency of the IIBL methods was reflected in the number of demonstrations participants gave to the robots equipped with different instance-retrieval methods. The average number of demonstrations given by the participants was: Robot A (m= 21.17, σ=6.44), Robot B (m= 29.17, σ=10.25), and Robot C (m= 24.15, σ=8.72). On average, participants provided 37.79% less demonstrations to robots utilizing IIBL retrieval methods (Robot A) than robots using k-NN (Robot B), while the average performance of the robots using IIBL was still better than that of the ones using k-NN. In the questionnaire asking when the participants stopped teaching each robot, majority of the participants answered "when Darwin cleared each level several times" for Robot A (63.64%) and Robot B(60.61%), and "when Darwin stopped improving" for Robot C (51.52%). Figure 61 depicts such participant responses as well as the average number of demonstrations given to each robot. Participants also spent almost twice (90.43%) more time with Robot B than Robot A, and 25.87% more with Robot B than Robot C. Participants spent more time instructing the robot when the robot was improving slower (Robot B, k-NN), but quickly stopped teaching when the robot wasn't responding to the demonstrations (Robot C).

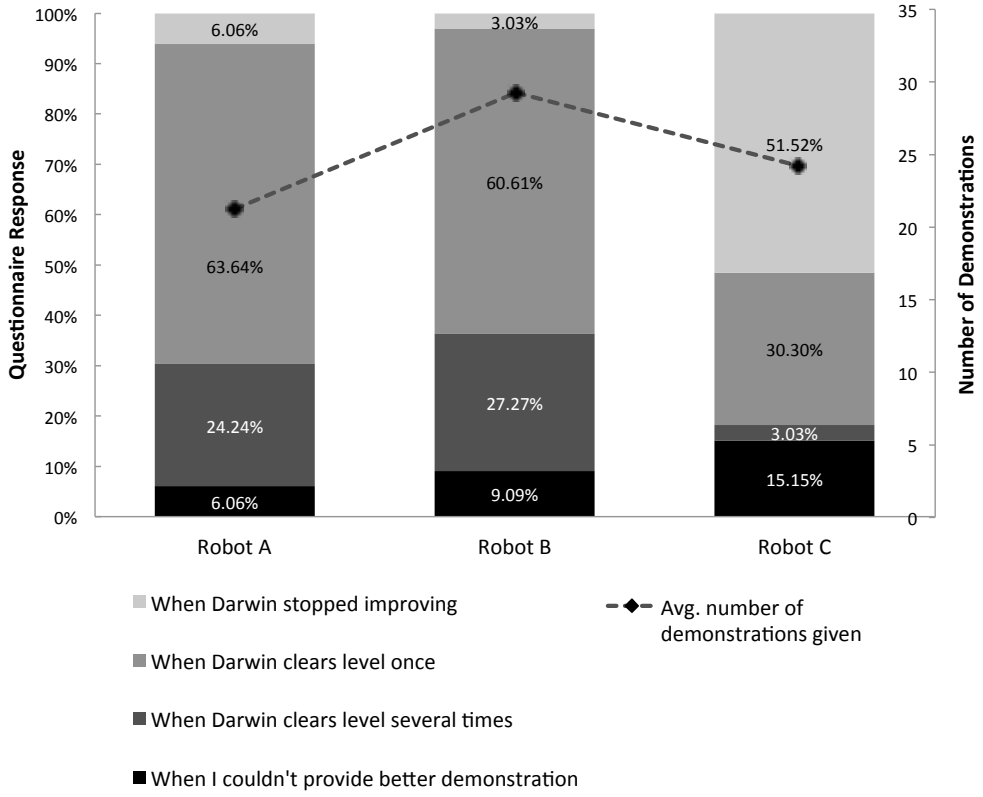


Figure 61: Participant responses to when they stopped teaching each robot and the average number of demonstrations given.

It was also observed that participants' teaching strategies changed over time (Figure 62). At the beginning, a group of participants either tried to give sufficient demonstrations to complete the whole game (batch demonstration, 21.21%) or at least a segment of each level (segmented batch demonstration, 30.30%). Another group first concentrated on teaching the robot the concept of shooting the bird by manipulating the touchscreen, and checking if the robot learnt the behavior after a few demonstrations (36.36%). By the later half of the session with the first robot, majority of the participants (42.42%) focused on improving robot's performance, therefore giving demonstrations only when the robot was struggling. When it was time to teach the second robot, less participants attempted to provide batch demonstrations at the beginning and spent more time receiving feedback from the robot. During the later half

of the session with the second robot, participants showed similar behavior as when teaching the first robot.

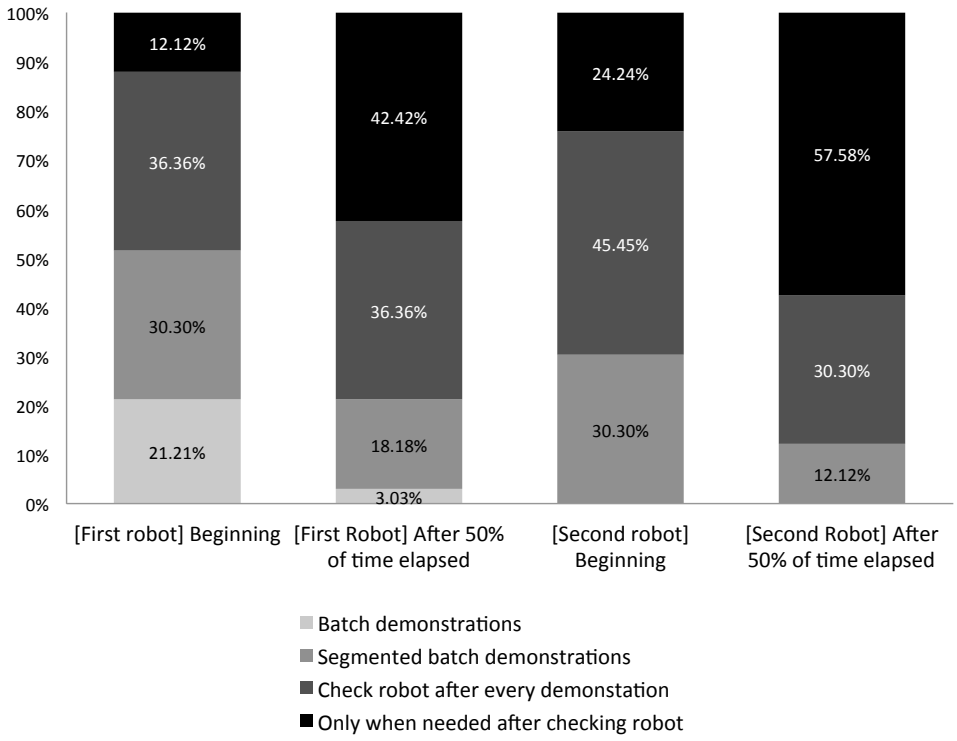


Figure 62: Change in teaching strategies at the beginning of an interaction and after 75% of the time had elapsed. Participants gradually transition from batch-mode demonstrations to just-in-time demonstrations.

Through these results, we observed that the participant's behavior adapted, e.g., the amount of interaction, decision wwhen to end the interaction, and the style of providing demonstrations, based on the robot learner's learning ability and performance.

6.5.2 Evaluation IV: Emergence of Social Behaviors

In general, it was observed that the participants utilized other forms of natural interactions though the robot only could learn from actual demonstrations of launching a bird. In this section, we validate our fourth hypothesis: Robot learners encourage participants to initiate more interaction than with another person. The natural forms

of interaction were measured as the length of time when an eye contact was made or when verbal/gestural-interaction behaviors were observed. These interactions were then categorized into instructive and non-instructive interactions. On average, participants spent 5 minutes and 42 seconds without the robot and 24 minutes and 5 seconds with the robot playing the game (Table 15). During these times, the ratios of the duration of social interactions initiated by the participants are measured to evaluate the significance of the robot effect. While less than 10% of the time was used to initiate conversation or eye contact with the experimenter, 34.81% of the time was used to initiate interaction with the robot.

Table 15: Average duration of interaction with the experimenter and with the robot.

with Experimenter	All Participants Combined	Child-female	Child-male	Child-combined
Avg. duration of interaction (min)	5.70	9.58	8.35	8.80
Avg. duration of initiated social behaviors (min)	0.18	0.66	0.51	0.56
Percentage of social behavior	3.28%	6.84%	6.13%	6.42%

with Robot	All Participants Combined	Child-female	Child-male	Child-combined
Avg. duration of interaction (min)	24.08	29.75	24.56	26.47
Avg. duration of initiated social behaviors (min)	8.38	13.60	8.31	10.26
Percentage of social behavior	34.81%	45.71%	33.83%	38.75%

Detailed categories of the interactions are listed in Table 16. Note that these cues are often observed simultaneously with one another, and the measurement ratio is calculated against the total time of the interaction. Women initiated more eye contact and conversation with the robot than men, 24.52% and 39.96% more respectively, while men used more gestural behavior than women (15.62%). It was also observed that women used more verbal, and men used more gestural behavior to instruct the robot.

Table 16: Categorized social behavior exhibited towards the robot.

Social Behaviors	Female	Male	Combined
Eye contact / Gaze	38.49% (of total duration)	30.91%	34.05%
Gestural interaction	14.85%	17.17%	16.21%
Verbal interaction	29.21%	20.87%	24.32%
Instructive	42.67%	33.42%	36.50%
Non-instructive	57.33%	66.58%	63.50%

Following each session, participants were asked a number of questions concerning their interaction experience with the robot (Figure 63). On a 5-point Likert scale, from strongly disagree (1) to strongly agree (5), the post-experiment survey reports that the participants felt their robot was socially interacting with them (m=4.7); was socially communicating with them (m=3.72); thought Darwin was learning from them (m=4.33); teaching Darwin was similar to their friends (m=4.01); and thought the robot enhanced their overall experience with the task (m=4.8). The survey form is attached in the Appendix B.

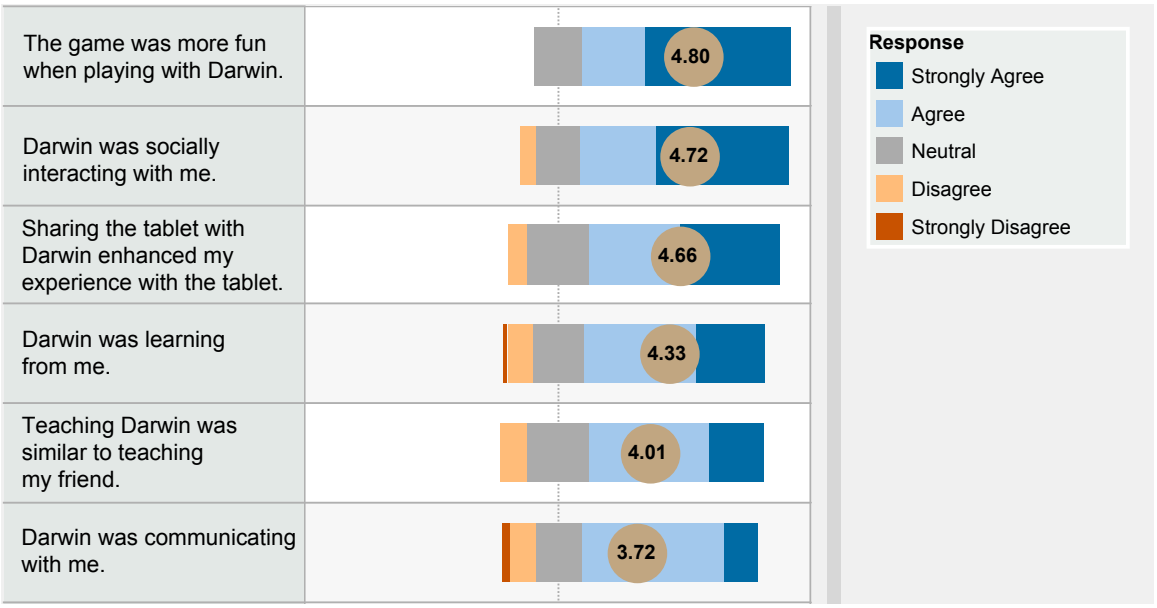
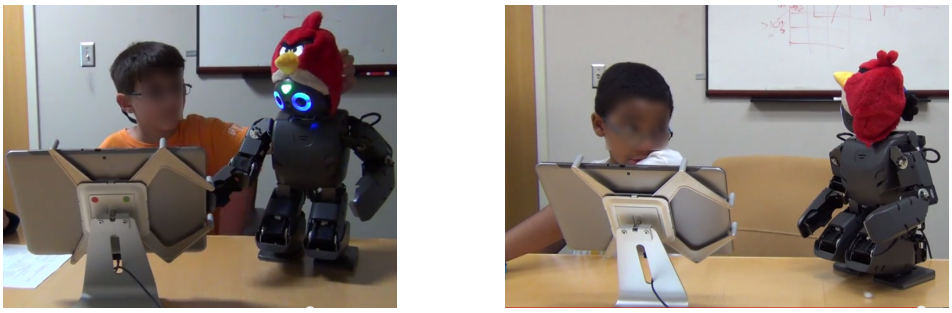


Figure 63: Post-experiment survey reports that the participants felt their robot was socially interacting with them and enhanced their overall experience with the task. (Graph depicted in 5-point Likert scale, from strongly disagree (1) to strongly agree (5))

6.5.3 Evaluation V: Pilot Study with ASD Children

Here, we report notable findings from a pilot study with two high-functioning children with autism spectrum disorder (ASD) compared to the typically developing children's group we have evaluated in the previous section. Avoiding interaction with other people is one of the key characteristics observed in children with ASD. The first participant (male, age 9), Figure 64(a), demonstrated close to average occurrences of social behaviors when the robot was present compared to the typically developing group. Out of 12.25 minutes spent in Session I, this child initiated interactions with the experimenter 2.87% of the time, which was little less than half the time (45%) the typically developing group had spent. In Session II, he spent 24.33 minutes with the robot and used 35.12% of the time interacting and instructing the robot. It was 91% of the time the combined comparison group had spent and 3.78% more than the boy's group alone. The observed behaviors were: eye contact (28.23%), gestural interaction



(a) (b)

Figure 64: A pilot study with two ASD children was conducted to evaluate robot learner's effect in encouraging social interaction.

(12.17%), and verbal interaction (28.90%). Table 17 summarizes the findings.

The second subject (male, age 6), Figure 64(b), eagerly participated in the task but did not initiate any interaction with the experimenter during Session A. He was excited about the robot's presence in Session B, but was apprehensive about engaging in the task due to misunderstanding of the robot's facial expression and the autonomous behavior. He spent most of the session (7.56 min) observing the robot and talking to himself, but also talking to his parent about the robot (28.14%). Though his interaction wasn't aiming toward the robot, the robot's behavior mediated a conversation with his parent and demonstrated close to 73% of the average time of the comparison group.

6.5.4 Discussion

In this study, our goal was to assess participants' acceptance and behavior towards an interactive robot learner. We examined how participants adapt their teaching behavior to robots with varying learning strategies and performances. Participants adjusted the number of demonstrations and the length of interaction that was needed to instruct each robot to fully learn the task. Based on the learning behavior of the robot, participants switched their timing when to provide demonstration and when to stop.

Table 17: ASD child participant and typically developing children's average duration of interaction with the experimenter and with the robot.

with Experimenter	Child-female	Child-male	Child-combined	ASD Child Participant
Avg. duration of interaction (min)	9.58	8.35	8.80	12.25
Avg. duration of initiated social behaviors (min)	0.66	0.51	0.56	0.35
Percentage of social behavior	6.84%	6.13%	6.42%	2.87%

with Robot	Child-female	Child-male	Child-combined	ASD Child Participant
Avg. duration of interaction (min)	29.75	24.56	26.47	24.33
Avg. duration of initiated social behaviors (min)	13.60	8.31	10.26	8.54
Percentage of social behavior	45.71%	33.83%	38.75%	35.12%

The second part of the study showed that the participants were motivated to teach the robot, and this process naturally fostered the emergence of social behaviors. Participants more actively initiated interaction with the robot than with another person. The type of interactions were mainly divided into instructing the robot and observing the robot's feedback. The human teachers utilized eye contact, gaze, verbal- and gestural behaviors to interact with the robots, and we observed some

gender difference in the form of interaction that is used the most. Women used more verbal instructions while men provided more demonstrations. Post-experiment survey indicated that the participants felt the robot learners were socially interacting with them and that the robots improved their experience with the task.

The pilot study with children with ASD also provided some preliminary evidence in understanding both the limitations of the system, as well as those attributes that are essential for establishing long-term interaction for engagement. We observed a child with ASD showing much less interaction with the experimenter, but exhibiting close to an average initiations of social interactions towards the robot, compared to typically developing children. Another child started an active discussion about the robot's behavior with his parents.

In regard to the overall experience, participants picked the IIBL robot learner as the most reliable and easiest to teach, but no significant correlation was found in the level of enjoyment among the progressive learning robots (Table 18). Other responses from the participants during an open-ended interview is attached in Appendix C.

Table 18: Post-experiment survey on participant's experience with each robot learner.

	Robot A (IIBL)	Robot B (k-NN)	Robot C (random)
was easiest to teach	69.70%	30.30%	0
was most reliable	78.79%	21.21%	0
was most enjoyable	39.39%	45.45%	15.15%

6.6 Summary

This chapter presented a novel approach that couples a robot learner with a tablet that functions as a shared workspace. The goal of this research was to design a

domain-independent learning system for a robot that continuously motivates engagement of the user. One of the limitations of commercially available robots is that they fail to provide new content when the user wants. We believe that continuous motivation comes from a continuous supply of new materials, and tablets provide such an environment. However, though tablets are a popular educational, therapeutic, and entertainment platform, people rarely exhibit social behaviors when interacting with the device themselves. By coupling a robot learner with a tablet, we observed the social behaviors emerging from the participants during an interaction of teaching a task to the robot.

The experiments were designed to validate our hypothesis regarding the performance of the system and the user experience. Evaluations I and II involved measuring the performance of the IIBL robot learner applied to a tablet shared workspace. Evaluation III explored how the teacher's behavior changed adapting to the robot's learning behavior and performance. Evaluation IV assessed participants' use of social cues during interaction with the robot learner. Evaluation V involved a pilot study with children with ASD. Before the evaluations, we conducted two pilot studies to reveal what features people focus on when they were describing the task to others. LWR and NNSA feature-weight prediction methods were applied and compared. Instances were successfully acquired through interactive demonstrations on a shared tablet workspace, which quantized the human teacher's gestural information into a state and action pair that becomes stored in memory. Interactive approaches were utilized to model the IIBL system, and two different feature sets recommended by the participants were used for evaluations. There were no significant performance differences between the two feature sets which shared three common features and two distinct features. It was observed that participants gradually changed their timing of providing demonstrations to only when the robot was struggling. Though interactive learning may not be the most efficient way to acquire a lot of instances at a short

amount of time, it provides an environment to effectively fill in the case base with necessary knowledge that the robot has not explored yet. Interactive learning also has shown to improve teacher's experience of the overall task as well as programming the robot.

Throughout the chapter, it was revealed that the tablet provided an intuitive environment for task engagement with the robot. Robots equipped with IIBL framework proved their effective and efficient performance and successfully created an interactive environment for interaction with human teachers. Future efforts will focus on enhancing the autonomy of the system such that the gaming App adapts in direct correlation to adaptation of the robot's social behaviors. This will ensure that both components correlate and grow with the capabilities of the child, as well as ensure the system is continuously engaging. Also, we would like to broaden our work with children with disabilities. As part of this future work, we will also study how the aspects of the robot (movement, sound, emotion expression) might affect interaction.

CHAPTER VII

CONCLUSIONS AND FUTURE WORK

The key motivation and contribution of this dissertation research has been in designing robot learners that accumulate task knowledge through interactive methods, and reusing the experience to engage with users in tasks conducted in a shared workspace. If there are many ways to accomplish a task, the robot learner's behavior customizes to that of the teacher's since the robot generates a solution space that was spanned by the demonstrations given by the teacher. We have highlighted the transparent reasoning characteristic of instance-based learning and the large tolerance for modeling task features as complement to interactive learning. Interactive methods reduce the problem of encoding knowledge and facilitates the process of acquiring instances of the task. The idea behind the IIBL framework is that if the system has a method of *extracting key features* that comprises a task and a set of *distance functions* to compute similarity between each feature, an instance-based knowledge for learning can be acquired independent of the task. We also presented regression and sensitivity measure methods to evaluate the contributing degrees of each feature variables on the prediction result to be used towards designing a retrieval function. We have shown results from comparison experiments to validate the effectiveness and the efficiency of IIBL.

Further research questions have been raised regarding the user's interaction with robot learners. We learned that the teacher's and the learner's behavior influence each other. The interaction time and the teacher's decision of when to end and when to provide examples are influenced by the performance and the progress of the learner. We also have derived conclusions that robot learners can have a positive

impact on encouraging the emergence of social behavior. Using the length of behaviors initiated by the participants as an indicator for the level of engagement, we have seen that the amount of social behavior significantly increased when the participants were interacting with robot learners, including children with ASD.

In the following, the contributions of this dissertation research are summarized, and the list of publications that resulted from this work is presented. Lastly, recommendations for future work is outlined.

7.1 Contributions

7.1.1 Interactive Instance-based Learning

Motivated by the methods of interactive machine learning and learning from demonstration, we have developed an embodied instance-based framework on a humanoid robot learner that achieves learning from interaction with humans in a shared workspace. Instance-based learning methods are usually data intensive, and the process of constructing a database requires expert knowledge and is often separated from the actual system deployment. The human-robot teaching and learning scenarios provide a natural environment for the teacher to endow the learner with the knowledge of key features that should be extracted from the scene in order to understand the task. The demonstrations, i.e., pairs of task states and actions, provided by the teacher are converted into instances that automate the process of case-base accural. By observing the learner's execution of the task, the teacher gains an idea of what instances should be enhanced, thereby making the acquisition process efficient and interactive.

The ability to retrieve and reuse prior experience as well as creating an environment for interactive learning are both important issues in building a task-independent learning framework for robots that serve different people's needs. The evaluation results showed that IIBL methods provide effective and efficient learning even for tasks that were not suitable for complete mathematical modeling.

7.1.2 Feature Weighting of Task Variables

We have presented two feature-weighting methods for designing a retrieval distance function and computing similarity between instances. The two methods were multivariate linear regression and an approach using sensitivity analysis with a fully connected feedforward neural network. The results showed that the solutions retrieved by both methods produce better performance and are more stable than those retrieved by a traditional k-NN regardless of k.

Increasing the explanatory behavior of a learning algorithm is often an issue in machine learning. When an instance-based approach is combined with rule-based methods, it can provide a comprehensible knowledge about how it arrived at a given prediction. By making the reasoning process transparent, IIBL further facilitates the teacher's understanding of the robot's behavior. A hybrid system of IIBL and ANN successfully modeled nonlinear behaviors of the given tasks. The IIBL retrieves prior instances based on the similarity measure recommended by the ANN, and triggers ANN to recommend a new set of feature weights when the human teacher tries to steer the system, i.e., increases the number of demonstrations judging from the robot's behavior, in a different direction.

Secondary contribution of feature weighting is in increasing the tolerance of acceptable user demonstrations. Demonstrations given by the user cannot be consistently successful. Even when the retrieved solution is beyond an acceptable threshold, instead of rejecting possible solutions, IIBL prefers to make mistakes that encourages the human user to observe the robot behavior and provide improved demonstrations at the right timing and enhance the quality of the case base.

7.1.3 Shared Workspace

The presence of both the teacher and the learner in a shared space makes learning more interactive. In Chapter 5, the task was conducted in a shared play scene

with blocks, and in Chapter 6, a novel approach to utilizing commercially available touchscreen tablets with robot learners was presented. By resolving the limitations of perceptual uncertainty and human-behavior modeling, a shared tablet workspace provided a robust environment for evaluating the utility of IIBL methods and conducting human-robot interaction studies. Furthermore, an endless support of context provided by these state-of-the-art devices encourage robots becoming long-term educational, entertainment, and therapeutic companions.

7.1.4 Encoding and Extracting Task Features

We developed a mechanism to incorporate human intuition into the task representation and feature-selection problems. Encoding instances requires many attributes of the task variables, e.g., what to extract from the task scene, the extraction method, the data type of the feature, and the similarity function to measure the distance between two features. In an interactive learning setting, we have observed human teachers using the keywords that represent feature knowledge. We provided the users with an interface to let the robot know what features possibly need to be learnt to understand the task. The selection of these features are not required to be precise since the IIBL system assigns weights to these features when computing similarity to maximize the system performance. Experiments showed that systems trained with different sets of features performed equally well.

In Chapter 3 and Chapter 5, we have proposed and evaluated some feature-extraction methods for object-based play scenarios. We have developed methods to model general manipulation behaviors of the play objective and to extract the task's goal using HMMs. A fast scale-invariant shape detection algorithm has also been proposed to facilitate the process of object matching. Further contributions have been made as we have applied the online training of task primitives to extract and recognize users' inputs through an assistive input device, TabAccess.

7.1.5 Experimental Findings about Teaching a Robot Learner

We analyzed how the teaching behavior adapts to the learner's learning progress and performance. It was observed that the total length of teaching depended on how much demonstration was provided, and the robot that was less efficient, i.e., required more demonstrations, encouraged elongated interaction. However, the robot that didn't demonstrate progressive learning had no positive impact on the length of interaction. We also measured the timing of when participants ended teaching and when they provided demonstrations. For the robots that were progressively learning, most participants ended teaching when the robots successfully completed each task at least once. For the robots that randomly retrieved instances, teaching stopped when the teachers realized the learning wasn't improving. As learning progressed, most of the participant's method of providing demonstrations transitioned from full/partial batch method to just-in-time intervention. Some participants first focused on teaching the primitive goal of the task, e.g., shooting a bird, and these participants checked the robot's performance after each demonstration. Later on when the learning had progressed, these participants also transitioned to providing just-in-time teaching.

The interaction with the robot learner positively encouraged the emergence of social behaviors in participants. More initiations of interaction were observed when participants interacted with the robot learner compared to when they were interacting with another person. From a qualitative pilot study with ASD children, we observed greatly improved rate of social behavior occurrences when interacting with the robot learner, comparable to typically developing children's group.

7.2 Publications

The following refereed publications were derived from this dissertation research.

1. Hae Won Park, Rick Coogle, and Ayanna M. Howard. Using a Shared Tablet Workspace for Interactive Demonstrations during Human-robot Learning Scenarios. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on.*

2. Hae Won Park and Ayanna M. Howard. Engaging Children In Social Behavior: Interaction with a Robot Playmate through Tablet-based Apps. In *Rehabilitation Engineering and Assistive Technology Society of North America Conference (RESNA), 2014.*

3. Ayanna M. Howard, Hae Won Park. Using Tablet Devices to Engage Children with Disabilities in Robotic Educational Activities. *Journal on Technology and Persons with Disabilities*, 2014.

4. Sergio Garcia-Vergara, LaVonda Brown, Hae Won Park, and Ayanna M. Howard. Engaging children in play therapy: The coupling of virtual reality (VR) games with social robotics. In *Technologies of Inclusive Well-Being.* Springer Berlin Heidelberg, 2014

5. Hae Won Park and Ayanna M. Howard. Interactive Instance-based Learning and Its Application to Therapeutic Tasks. Late breaking report, In *Human-Robot Interaction Pioneers Workshop (HRI), 2014 ACM/IEEE International Conference on.*

6. Ayanna M. Howard, Sergio Garca-Vergara, LaVonda Brown, and Hae Won Park. Engaging Children in Rehabilitation through Virtual Reality Robot-Assisted Therapy Approaches. In *IROS 2013 Workshop on Healthcare Robotics and Wearable Systems*, 2013.

7. Hae Won Park and Ayanna M. Howard. Tabaccess, a Wireless Controller for Tablet Accessibility for Individuals with Limited Upper-body Mobility. In *ISSNIP*

Biosignals and Biorobotics Conference (BRC), 2013 IEEE International Conference on.

8. Hae Won Park and Ayanna M. Howard. Providing Tablets as Collaborative-task Workspace for Human-robot Interaction. Late breaking report in *Human-Robot Interaction (HRI), 2013 ACM/IEEE International Conference on.*

9. Luke Roberts, Hae Won Park, and Ayanna M. Howard. Robots and Therapeutic Play: Evaluation of a Wireless Interface Device for Interaction with a Robot Playmate. In *Engineering in Medicine and Biology Society (EMBC), 2012 IEEE International Conference on.*

10. Hae Won Park. Task-learning Policies for Collaborative Task Solving in Human-robot Interaction. Extended Abstracts, In *Multimodal interaction (ICMI), 2012 ACM International Conference on.*

11. Hae Won Park and Ayanna M. Howard. Understanding child's play by sequencing play primitives and planning turn-taking strategy for a therapeutic robot playmate. In *Pediatric Research Retreat: Frontiers in Pediatric Science, 2012.*

12. Hae Won Park and Ayanna M. Howard. Case-based Reasoning for Planning Turn-taking Strategy with a Therapeutic Robot Playmate. In *Biomedical Robotics and Biomechatronics (BioRob), 2010 IEEE RAS and EMBS International Conference on.*

13. Hae Won Park and Ayanna M. Howard. Understanding a Child's Play for Robot Interaction by Sequencing Play Primitives using Hidden Markov Models. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on.*

14. Ayanna M. Howard, Sekou Remy, Chung Hyuk Park, Hae Won Park, and Douglas Brooks. Intelligent robotics for assistive healthcare and therapy. *The Path to Autonomous Robots*, Springer Press, 2009.

15. Alex J.B. Trevor, Hae Won Park, Ayanna M. Howard, and Charles C. Kemp. Playing with Toys: Towards Autonomous Robot Manipulation for Therapeutic Play. In *Robotics and Automation (ICRA), 2009 IEEE International Conference on.*

16. Ayanna M. Howard, Sekou Remy, and Hae Won Park. Learning of Arm Exercise Behaviors: Assistive Therapy based on Therapist-Patient Observation. In *RSS: Workshop on Interactive Robot Learning, Zurich, Switzerland, 2008.*

17. Ayanna M. Howard, Hae Won Park, and Charles C. Kemp. Extracting Play Primitives for a Robot Playmate by Sequencing Low-level Motion Behaviors. In *Robot and Human Interactive Communication (RO-MAN), 2008 IEEE International Symposium on.*

In addition to refereed research publications, this dissertation work was recognized in several international and national competitions. The demonstration of "Angry DARwIn: Framework for Human-robot Task Collaboration on a Shared Tablet Workspace" took first place at the Humanoid Applications Challenge at the 2013 IEEE International Conference on Robotics and Automation (ICRA 2013), and received the Best Video Award at the same venue. The project "TabAccess, a Wearable Tablet Interface for Individuals with Motor Impairment" was one of the finalists at the Cornell Cup sponsored by Intel in 2012. Two patents were filed and published from the technologies developed during this research: "Methods, controllers, and computer program products for accessibility to computing devices" (International publication number WO 2013/166261) and "Method and system for facilitating interactions between robots and children using a shared tablet-based interface" (USPTO serial number 61/897,406).

7.3 Recommendations for Future Work

Instance Adaptation and Task Categorization: This research dealt with generalizing task-independent learning through acquiring task-specific knowledge from the interactive methods. However, the problem of adapting instances retrieved from prior instances was difficult to generalize since the process requires expert knowledge that is beyond the intuition of most people. The adaptation methods used in this research included substituting some features with what is currently available in the scene (Chapter 5) and averaging over k number of retrieved instances weighted by their distance to a given problem (Chapter 6).

Adaptation is a major challenge in instance-based methods [53, 112, 129]. Prior works suggest substitutional, transformational, and generative methods to adaptation [70, 81, 95, 102, 128]. Substitutional adaptation exchanges the values of some solution attributes; transformational approaches modify the structure of the solution to include or remove features of the retrieved instances to satisfy the requirements of the current problem; and generative methods produce a new solution from the problem features with a predefined procedure that immediately generates a solution from a given query. A hybrid approach of these methods could be utilized for categorizing tasks, possibly by using the task features as guidance to training a categorization model.

Conveying Robot Capabilities to the Teachers and Encouraging Better Teaching: Social robots are gaining more attention as many robots seek to work alongside humans. It is now not only important for the robot to understand a human's actions and intentions, but we should also develop ways for the human partner to gain better understanding of the robot's capabilities, how it processes the environment, and the reasoning behind its actions. This dissertation research contributed to this goal by making the reasoning process transparent using an instance-based approach,

but the method for making the process more accessible requires further work.

Prior to introducing the task-feature encoding interface to the participants, a brief description of the robot's available skills were given. Through this process, we have identified the following problems: 1) The robot's skills cannot be intuitively recognized by the participants, 2) The robot's appearance deceives its capabilities, and 3) the conceptual keywords used for describing the task features varies greatly in its meaning across tasks. Also, until now, the system fully depended on the teacher's discretion when to end the teaching. It is an NP-hard problem to reason whether the problem space has been covered well, but by taking the approach to detect failure, the system can provide the user with better knowledge of how much the system has learnt.

We have also attempted to make the IIBL system more tolerant to the quality of the user demonstrations by retraining the feature weights when needed, but there are situations when a more aggressive method should be taken to gain better demonstrations. There have been prior works that use reinforcement learning and active learning methods to resolve the uncertainty in human teaching [30, 33, 48, 49, 74]. Further advancements could be made to inform the teacher of the current state of the robot, i.e., resolving the clarity of conflicting inputs, requesting instances for the unexplored problem space, and asking to add/remove task features that will help generalize the task model better.

Participant Trials with Children with Special Needs: Finally, we evaluated the ability of the robot learner to evoke social behavior in children. Further work involving additional trials would allow us to push this technology for adoption in clinical settings.

APPENDIX A

PROTOCOL FOR ANGRY DARWIN EXPERIMENTS

Experiment Protocol for Angry DARwIn

0. Apparatus
    a. Participant recruitment: During robot learner demonstrations in various events including open house, exhibitions, and technology demonstrations, a sign-up sheet was used to obtain e-mail addresses of the attendees (or the parents of the attendees) interested in participating in the study. This information was used to solicit participants.
    b. DARwIn-OP is used as a robot learner.
    c. Two Galaxy Tabs are mounted on tablet stands.
    d. Data recording: Two video camcorders - one in front capturing human-robot interaction and the other at the back capturing the tablet screen events.

1. Step 1
    a. Instruction: "This is a game similar to Angry Birds. You shoot the bird to destroy all the pigs. There are four levels in total, and the difficulty gradually increases. I wasn't able to solve all levels, but you might. I will be sitting right next to you, so show me how you play, give me any comments, or ask any questions. Please start playing the game, try all levels, and when you feel like you're done playing, just let me know."
    b. The experimenter sits next to the participant.
    c. Record the interaction.
    d. Measure time.

2. Step 2
    a. In step 2, the participants are divided into four groups:

        Robot A: The robot learner that bases his learning on IIBL. This method uses the sum of Gaussian weighting on four best matching solutions. The feature weights are trained using IIBL methods.

        Robot B: The robot learner that bases his learning on *k*-NN. This method uses the sum of Gaussian weighting on four best matching solutions. The feature weights are equally distributed.

        Robot C: The robot learner that randomly retrieves a solution from the case base.

        i. Group 1: Robot A + Robot B
        ii. Group 2: Robot B + Robot A
        iii. Group 3: Robot A + Robot C
        iv. Group 4: Robot C + Robot A

    b. Instruction: "Now, I'd like you to teach Darwin to play the same game. Just teach him in the same manner you would teach your friend. Provide Darwin with demonstrations on how to solve each level. Whenever you reach out to provide a demonstration to Darwin, he will wait for his turn. Continue teaching each level until you are satisfied that Darwin has learned the level well enough, or you think Darwin has stopped learning. Later, I want you to show me what you have taught

Darwin, and collaboratively solve each level with him. Darwin may or may not try to communicate with you, and he may not use human language. Afterwards, I will ask you some questions about your experience teaching a task to Darwin."

c. Record the interaction in two angles.
d. Measure time.

3. Open-ended interview
    a. What was Darwin trying to communicate with his gestures and voice?
    b. What was the difficulty of teaching each level?
    c. How many times do you think in average you provided demonstration?
    d. How well do you think Darwin learned each level?
    e. What was your general teaching strategy?
    f. When did you decide you interrupt Darwin and provide more demonstrations?
    g. When did you decide to stop each teaching session?
    h. What else would you have liked to communicate?
    i. How would you evaluate Darwin as a collaborator?
    j. What was you strategy for collaboration?
    k. What would you expect from Darwin as a collaborator?
    l. What other collaborative task would you like to conduct with a robot?

APPENDIX B

SURVEY FOR ANGRY DARWIN EXPERIMENTS

Participant Experience Survey

Date: _____ Age:_____ Female / Male

I. Learning and Collaborating – Robot without the Hat -

1. In a scale of 1-5, what was the difficulty of teaching each level? (1: Easiest - 5: Hardest)
   Level 1: _____ Level 2: _____ Level 3:_____ Level 4: _____

2. How many times do you think in average you provided demonstration for each level?
   Level 1: _____ Level 2: _____ Level 3:_____ Level 4: _____

3. What was your general teaching strategy?
   (a) Provide successful demonstrations of all levels before checking Darwin's performance.
   (b) Provide several demonstrations in each level before checking Darwin's performance.
   (c) Check Darwin's performance after every demonstration.
   (d) Check Darwin's performance before providing any demonstration.

4. When did you decide to stop teaching each level?
   (a) When Darwin seemed like he was not learning.
   (b) When Darwin's performance was no longer improving.
   (c) When Darwin cleared the level once.
   (d) When Darwin cleared the level several times.
   (e) When the level was too hard and neither I nor Darwin could clear the level.

5. How would you evaluate Darwin as a collaborator?
   (a) I would rely on Darwin to solve the task from start to finish.
   (b) I would mostly rely on Darwin to solve the task, but will interrupt when Darwin's progress is slow.
   (c) I would take turns solving the task with Darwin and equally contribute.
   (d) I'd only want Darwin to help when I get stuck.
   (e) I would rather solve the task by myself.

6. Please rate each statement.

1	2	3	4	5
I strongly disagree	I disagree	Uh… not sure	I agree	I strongly agree

Darwin is an efficient learner.	
Darwin is a slow learner.	
Darwin has learned well.	
Darwin was able to recognize good and bad examples and learn from the good ones.	
Darwin was randomly shooting the bird.	
Darwin provided consistent performance.	

II. Learning and Collaborating – Robot with the Hat -

1. In a scale of 1-5, what was the difficulty of teaching each level? (1: Easiest - 5: Hardest)
   Level 1: _____ Level 2: _____ Level 3:_____ Level 4: _____

2. How many times do you think in average you provided demonstration for each level?
   Level 1: _____ Level 2: _____ Level 3:_____ Level 4: _____

3. What was your general teaching strategy?
   (a) Provide successful demonstrations of all levels before checking Darwin's performance.
   (b) Provide several demonstrations in each level before checking Darwin's performance.
   (c) Check Darwin's performance after every demonstration.
   (d) Check Darwin's performance before providing any demonstration.

4. When did you decide to stop teaching each level?
   (a) When Darwin seemed like he was not learning.
   (b) When Darwin's performance was no longer improving.
   (c) When Darwin cleared the level once.
   (d) When Darwin cleared the level several times.
   (e) When the level was too hard and neither I nor Darwin could clear the level.

5. How would you evaluate Darwin as a collaborator?
   (a) I would rely on Darwin to solve the task from start to finish.

(b) I would mostly rely on Darwin to solve the task, but will interrupt when Darwin's progress is slow.
(c) I would take turns solving the task with Darwin and equally contribute.
(d) I'd only want Darwin to help when I get stuck.
(e) I would rather solve the task by myself.

6. Please rate each statement.

1	2	3	4	5
I strongly disagree	I disagree	Uh… not sure	I agree	I strongly agree

Darwin is an efficient learner.	
Darwin is a slow learner.	
Darwin's learned well.	
Darwin was able to recognize good and bad examples and learn from the good ones.	
Darwin was randomly shooting the bird.	
Darwin provided consistent performance.	

7. Which robot was easier to teach?
(a) The robot without the hat.
(b) The robot wearing the hat.

8. Which robot was the most reliable performer?
(a) The robot without the hat.
(b) The robot wearing the hat.

9. Which did you enjoy the most?
(a) Only tablet.
(b) Tablet and the robot without the hat.
(c) Tablet and the robot wearing the hat.
(d) Tablet and the person.

10. Which did you enjoy the least?
(a) Only tablet.
(b) Tablet and the robot without the hat.
(c) Tablet and the robot wearing the hat.
(d) Tablet and the person.

III. Shared Workspace

1. Please rate each statement.

1	2	3	4	5
I strongly disagree	I disagree	Uh… not sure	I agree	I strongly agree

Sharing an event (playing angry birds) with Darwin enhanced my experience with a tablet.	
I thought the game was more fun when playing it with Darwin.	
Tablet provided good activities to conduct together with Darwin.	

2. What other collaborative task would you like to conduct with Darwin on a tablet?

IV. Interaction

1. What were the things that Darwin was able to do? (Select all that apply)
"Darwin was able to _____"
(a) Talk to me (b) Look at me
(c) Learn from me (d) Teach me better strategy
(e) Ask for help (f) Help me solve the level
(g) Express emotions (h) Generate his own strategy
(i) Take turns with me (j) Respond when I call him
(k) Listen to my comments

2. What do you think Darwin tried to communicate with his gestures and voice? (Select all that apply)
(a) "Yes/No"
(b) "Did you call me?"
(c) "Do you want me to play?"
(d) "I want to play."
(e) "Take your turn."
(f) "I liked/disliked your shot."
(g) "I have a better idea. I'll show you."
(h) "Can you teach me again?"
(i) "Congrats! You/we did it!"
(j) "That's sad."
(k) "Let's play again."
(l) "I want to stop playing."
(m) "I don't understand."
(n) "How was my shot?"
(o) "It's too easy/hard."

3. Please rate each statement.

1	2	3	4	5
I strongly disagree	I disagree	Uh… not sure	I agree	I strongly agree

Darwin was socially interacting with me.	
Teaching Darwin was similar to teaching my friend.	
I couldn't figure out what Darwin wanted.	
I would love to take Darwin home and play with him more.	

4. What else would you have liked to communicate?

APPENDIX C

INTERVIEW RESPONSES FOR ANGRY DARWIN EXPERIMENTS

Open-ended Interview Responses

In the following, Group A interacted with robots that produced progressive learning (Robot A and Robot B), and Group B interacted with a robot that generated behaviors based on a random selection from demonstrated instances (Robot C).

1. What was Darwin trying to communicate with his gestures and voice?
    a. He was trying to communicate his understanding.
    b. He was communicating his emotions: success and frustration.
    c. His eye contact was very good and I felt he was very attentive.
2. What was the difficulty of teaching each level?
    a. Group A: I sometimes provided bad examples, but it seems that the robot is able to neglect them when I provide the right demonstration.
    b. Group B: The first level was really hard because he didn't seem to learn and sometimes even shoot the bird backwards.
3. How many times do you think in average you provided demonstration?
    a. Group A: 2~3 times depending on how successful my demonstrations were. The last level took me about 6 demonstrations though.
    b. Group B: First level over fifteen, where I think 4~5 is sufficient for any level.
4. How well do you think Darwin learned each level?
    a. Group A: he learned each level very fast and very well. His performance is also very consistent.
    b. Group B: At first Darwin was totally dumb, seemed like it was programmed to output random behavior. Then later on, his performance was so good that I felt it was almost unnatural.
5. What was your general teaching strategy?
    a. Group A: I provided some successful demonstrations and let Darwin play and watched his performance to make sure he learned.
    b. Group B: Focused on teaching the very basic concept: "Shoot forward"
6. When did you decide you interrupt Darwin and provide more demonstrations?
    a. Group A: When his try wasn't good enough.
    b. Group B: When I thought he was lost or when he was very close but didn't succeed.
7. When did you decide to stop each teaching session?
    a. Group A: When he successfully destroyed all the enemies.
    b. Group B: When he seemed like he wasn't getting any better.
8. What else would you have liked to communicate?
    a. Group A: I wanted to let him know whether the demonstration I just gave was good or bad.
    b. Group B: Which part of the demonstration he doesn't understand.
9. How would you evaluate Darwin as a collaborator?
    a. Group A: He is definitely a good collaborator. I cannot provide consistent performance, but he can. I will rely on him more as the task gets more difficult.
    b. Group B: After a while, it seemed he finally learned to play, and provided consistent performance. I could see him as a collaborator if he can learn quicker.

10. What was you strategy for collaboration?
    a. Group A: I took turns shooting the bird. For more difficult task, I let him take most of the turns.
    b. Group B: It took a lot of time teaching Darwin to do the task, so I anticipate Darwin to take over the task and minimize my part.
11. What would you expect from Darwin as a collaborator?
    a. Group A: I want Darwin to remember the details and the sequences I might forget about a task, and instruct me when I'm making a mistake.
    b. Group B: Learn quicker and reduce my workload.
12. What other collaborative task would you like to conduct with a robot?
    a. Group A: Assembling furniture (an example the experimenter gave) sounds good.
    b. Group B: Cleaning.

Bibliography

[1] Aha, D. W., "The omnipresence of case-based reasoning in science and application," *Knowledge-based systems*, vol. 11, no. 5, pp. 261–273, 1998.

[2] Aha, D. W., Kibler, D., and Albert, M. K., "Instance-based learning algorithms," *Machine learning*, vol. 6, no. 1, pp. 37–66, 1991.

[3] Altman, N. S., "An introduction to kernel and nearest-neighbor nonparametric regression," *The American Statistician*, vol. 46, no. 3, pp. 175–185, 1992.

[4] Argall, B. D., Chernova, S., Veloso, M., and Browning, B., "A survey of robot learning from demonstration," *Robotics and Autonomous Systems*, vol. 57, no. 5, pp. 469–483, 2009.

[5] Ashley, K., "Reasoning with cases and hypotheticals in hypo," *International Journal of Man-Machine Studies*, vol. 34, no. 6, pp. 753–796, 1991.

[6] Atkeson, C. and Schaal, S., "Robot learning from demonstration," in *Machine Learning: Proceedings of the Fourteenth International Conference (ICML '97)*, pp. 12–20, 1997.

[7] Atkeson, C. G., Moore, A. W., and Schaal, S., "Locally weighted learning," *Artificial intelligence review*, vol. 11, no. 1-5, pp. 11–73, 1997.

[8] Baranek, G., Barnett, C., Adams, E., Wolcott, N., Watson, L., and Crais, E., "Object play in infants with autism: Methodological issues in retrospective video analysis," *American Journal of Occupational Therapy*, vol. 59, no. 1, pp. 20–30, 2005.

[9] Bauer, M. A., "Programming by examples," *Artificial Intelligence*, vol. 12, no. 1, pp. 1–21, 1979.

[10] Baxter, P., Wood, R., and Belpaeme, T., "A touchscreen-based sandtrayto facilitate, mediate and contextualise human-robot social interaction," in *Human-Robot Interaction (HRI), 2012 7th ACM/IEEE International Conference on*, pp. 105–106, IEEE, 2012.

[11] Bay, H., Tuytelaars, T., and Van Gool, L., "Surf: Speeded up robust features," *Computer Vision–ECCV 2006*, pp. 404–417, 2006.

[12] Bergmann, R., Kolodner, J., and Plaza, E., "Representation in case-based reasoning," *The Knowledge Engineering Review*, vol. 20, no. 03, pp. 209–213, 2005.

[13] Biggs, J., "Romo, The iPhone-Powered Robot, Grows Up," Oct. 16 2012.

[14] Bill, "iPad Controlled Robots Help Kids With Autism," Aug. 13 2010.

[15] Billard, A., Robins, B., Nadel, J., and Dautenhahn, K., "Building robota, a mini-humanoid robot for the rehabilitation of children with autism," *Assistive Technology*, vol. 19, no. 1, pp. 37–49, 2007.

[16] Billard, A. G., Calinon, S., and Guenter, F., "Discriminative and adaptive imitation in uni-manual and bi-manual tasks," *Robotics and Autonomous Systems*, vol. 54, no. 5, pp. 370–384, 2006.

[17] Bradski, G., "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[18] Bradski, G. and Kaehler, A., *Learning OpenCV: Computer vision with the OpenCV library*. O'Reilly Media, Inc., 2008.

[19] Breazeal, C. and Scassellati, B., "Challenges in building robots that imitate people," *Imitation in animals and artifacts*, p. 363, 2002.

[20] Breazeal, C. and Scassellati, B., "Robots that imitate humans," *Trends in cognitive sciences*, vol. 6, no. 11, pp. 481–487, 2002.

[21] Brooks, A., Gray, J., Hoffman, G., Lockerd, A., Lee, H., and Breazeal, C., "Robot's play: interactive games with sociable machines," *Computers in Entertainment (CIE)*, vol. 2, no. 3, pp. 10–10, 2004.

[22] Brooks, D. and Howard, A., "A computational method for physical rehabilitation assessment," in *Biomedical Robotics and Biomechatronics (BioRob), 2010 3rd IEEE RAS and EMBS International Conference on*, pp. 442 –447, Sept. 2010.

[23] Brunelli, R., *Template matching techniques in computer vision*. Wiley, 2008.

[24] Burgar, C., Lum, P., Shor, P., and Van der Loos, H., "Development of robots for rehabilitation therapy: The palo alto va/stanford experience," *Journal of Rehabilitation Research and Development*, vol. 37, no. 6, pp. 663–674, 2000.

[25] Bushey, T., "Autism Games," Jan. 2009.

[26] Cakmak, M. and Thomaz, A. L., "Designing robot learners that ask good questions," in *Proceedings of the seventh annual ACM/IEEE international conference on Human-Robot Interaction*, pp. 17–24, ACM, 2012.

[27] Calinon, S., Guenter, F., and Billard, A., "On learning, representing, and generalizing a task in a humanoid robot," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 37, pp. 286–298, April 2007.

[28] Canny, J., "A computational approach to edge detection," *Readings in computer vision: issues, problems, principles, and paradigms*, p. 184, 1987.

[29] Casby, M., "Developmental assessment of play a model for early intervention," *Communication Disorders Quarterly*, vol. 24, no. 4, pp. 175–183, 2003.

[30] Chao, C., Cakmak, M., and Thomaz, A. L., "Transparent active learning for robots," in *Human-Robot Interaction (HRI), 2010 5th ACM/IEEE International Conference on*, pp. 317–324, IEEE, 2010.

[31] Chen, D. and Burrell, P., "Case-based reasoning system and artificial neural networks: A review," *Neural Computing & Applications*, vol. 10, no. 3, pp. 264–276, 2001.

[32] Chernova, S., DePalma, N., and Breazeal, C., "Crowdsourcing real world human-robot dialog and teamwork through online multiplayer games," *AI Magazine*, vol. 32, no. 4, pp. 100–111, 2011.

[33] Chernova, S. and Veloso, M., "Multi-thresholded approach to demonstration selection for interactive robot learning," in *Human-Robot Interaction (HRI), 2008 3rd ACM/IEEE International Conference on*, pp. 225–232, IEEE, 2008.

[34] Cover, T. and Hart, P., "Nearest neighbor pattern classification," *Information Theory, IEEE Transactions on*, vol. 13, no. 1, pp. 21–27, 1967.

[35] Cuddihy, P. and Cheetham, W., "Elsi: A medical equipment diagnostic system," *Case-Based Reasoning Research and Development*, pp. 722–722, 1999.

[36] Darrell, T. and Pentland, A., "Space-time gestures," in *Computer Vision and Pattern Recognition, 1993. Proceedings CVPR '93., 1993 IEEE Computer Society Conference on*, pp. 335–340, June 1993.

[37] Fails, J. A. and Olsen Jr, D. R., "Interactive machine learning," in *Proceedings of the 8th international conference on Intelligent user interfaces*, pp. 39–45, ACM, 2003.

[38] Feil-Seifer, D. and Mataric, M., "Defining socially assistive robotics," in *Rehabilitation Robotics, 2005. ICORR 2005. 9th International Conference on*, pp. 465–468, IEEE, 2005.

[39] Fernández-López, Á., Rodríguez-Fórtiz, M., Rodríguez-Almendros, M., and Martínez-Segura, M., "Mobile learning technology based on ios devices to support students with special education needs," *Computers & Education*, 2012.

[40] Floyd, M. W., Esfandiari, B., and Lam, K., "A case-based reasoning approach to imitating robocup players.," in *FLAIRS Conference*, pp. 251–256, 2008.

[41] García-Vergara, S., Brown, L., Park, H. W., and Howard, A. M., "Engaging children in play therapy: The coupling of virtual reality games with social robotics," in *Technologies of Inclusive Well-Being*, pp. 139–163, Springer, 2014.

[42] Garson, G. D., "Interpreting neural-network connection weights," *AI expert*, vol. 6, no. 4, pp. 46–51, 1991.

[43] Gartner, A. and others, *Children Teach Children: Learning by Teaching*. ERIC, 1971.

[44] Gevrey, M., Dimopoulos, I., and Lek, S., "Review and comparison of methods to study the contribution of variables in artificial neural network models," *Ecological Modelling*, vol. 160, no. 3, pp. 249–264, 2003.

[45] Ginsburg, K., "The importance of play in promoting healthy child development and maintaining strong parent-child bonds.," *Pediatrics*, vol. 119, no. 1, pp. 182–91, 2007.

[46] Goh, A., "Back-propagation neural networks for modeling complex systems," *Artificial Intelligence in Engineering*, vol. 9, no. 3, pp. 143–151, 1995.

[47] Graf, B., Hans, A., Kubacki, J., and Schraft, R., "Robotic home assistant care-o-bot ii," in *[Engineering in Medicine and Biology, 2002. 24th Annual Conference and the Annual Fall Meeting of the Biomedical Engineering Society] EMBS/BMES Conference, 2002. Proceedings of the Second Joint*, vol. 3, pp. 2343–2344, IEEE, 2002.

[48] Gribovskaya, E., d'Halluin, F., and Billard, A., "An active learning interface for bootstrapping robot's generalization abilities in learning from demonstration," in *RSS Workshop Towards Closing the Loop: Active Learning for Robotics*, 2010.

[49] Grollman, D. H. and Jenkins, O. C., "Dogged learning for robots," in *ICRA*, pp. 2483–2488, 2007.

[50] Gupta, A., Park, S., and Lam, S. M., "Generalized analytic rule extraction for feedforward neural networks," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 11, no. 6, pp. 985–991, 1999.

[51] Ha, I., Tamura, Y., Asama, H., Han, J., and Hong, D. W., "Development of open humanoid platform darwin-op," in *SICE Annual Conference (SICE), 2011 Proceedings of*, pp. 2178–2181, IEEE, 2011.

[52] Hammond, K., *Case-based planning: viewing planning as a memory task*. Academic Press Professional, Inc., 1989.

[53] Hanney, K. and Keane, M. T., *Learning adaptation rules from a case-base*. Springer, 1996.

[54] Hecht-Nielsen, R., "Theory of the backpropagation neural network," in *Neural Networks, IJCNN 1989, International Joint Conference on*, pp. 593–605, IEEE, 1989.

[55] Hoffman, G. and Vanunu, K., "Effects of robotic companionship on music enjoyment and agent perception," in *Proceedings of the 8th ACM/IEEE international conference on Human-robot interaction*, pp. 317–324, IEEE Press, 2013.

[56] Howard, A., Park, H. W., and Kemp, C., "Extracting play primitives for a robot playmate by sequencing low-level motion behaviors," in *Robot and Human Interactive Communication, 2008. RO-MAN 2008. The 17th IEEE International Symposium on*, pp. 360–365, Aug. 2008.

[57] Howard, A., "Robots learn to play: Robots emerging role in pediatric therapy," *26th International Florida Artificial Intelligence Research Society Conference*, 2013.

[58] Hu, W., "Math that moves: schools embrace the ipad," *The New York Times*, vol. 4, 2011.

[59] IGN, "E3 2010: Project Natal is "Kinect"," June 13 2010.

[60] Iwai, Y., Watanabe, K., Yagi, Y., and Yachida, M., "Gesture recognition using colored gloves," in *Pattern Recognition, 1996., Proceedings of the 13th International Conference on*, vol. 1, pp. 662–666 vol.1, Aug. 1996.

[61] Jung, Y. and Lee, K. M., "Effects of physical embodiment on social presence of social robots," *Proceedings of PRESENCE*, pp. 80–87, 2004.

[62] Jung, Y., Park, H., Choi, Y., and Myaeng, S.-H., "Designing a cognitive case-based planning framework for home service robots," in *Robot and Human interactive Communication, 2007. RO-MAN 2007. The 16th IEEE International Symposium on*, pp. 827–832, IEEE, 2007.

[63] Kim, H. and de Araújo, S., "Grayscale template-matching invariant to rotation, scale, translation, brightness and contrast," *Advances in Image and Video Technology*, 2007.

[64] Kolodner, J. L., "An introduction to case-based reasoning," *Artificial Intelligence Review*, vol. 6, no. 1, pp. 3–34, 1992.

[65] Kolodner, J., Simpson, R., and Sycara-Cyranski, K., "A process model of case-based reasoning in problem solving," in *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, vol. 1, 1985.

[66] Kourtzi, Z. and Kanwisher, N., "Cortical regions involved in perceiving object shape," *The Journal of Neuroscience*, vol. 20, no. 9, pp. 3310–3318, 2000.

[67] Krebs, H. and Hogan, N., "Therapeutic robotics: A technology push," *Proceedings of the IEEE*, vol. 94, pp. 1727 –1738, Sept. 2006.

[68] Kristoffersson, A., Coradeschi, S., and Loutfi, A., "A review of mobile robotic telepresence," *Advances in Human-Computer Interaction*, vol. 2013, p. 3, 2013.

[69] Kronreif, G., Prazak, B., Mina, S., Kornfeld, M., Meindl, M., and Furst, M., "Playrob - robot-assisted playing for children with severe physical disabilities," in *Rehabilitation Robotics, 2005. ICORR 2005. 9th International Conference on*, pp. 193–196, 2005.

[70] Leake, D. B., *Case-based reasoning: Experiences, lessons and future directions*. MIT press, 1996.

[71] Lee, C. and Xu, Y., "Online, interactive learning of gestures for human/robot interfaces," in *Robotics and Automation, 1996. Proceedings., 1996 IEEE International Conference on*, vol. 4, pp. 2982–2987 vol.4, April 1996.

[72] Lee, K. M., Jung, Y., Kim, J., and Kim, S. R., "Are physically embodied social agents better than disembodied social agents?: The effects of physical embodiment, tactile interaction, and people's loneliness in human–robot interaction," *International Journal of Human-Computer Studies*, vol. 64, no. 10, pp. 962–973, 2006.

[73] Lek, S., Delacoste, M., Baran, P., Dimopoulos, I., Lauga, J., and Aulagnier, S., "Application of neural networks to modelling nonlinear relationships in ecology," *Ecological modelling*, vol. 90, no. 1, pp. 39–52, 1996.

[74] Lopes, M., Melo, F., and Montesano, L., "Active learning for reward estimation in inverse reinforcement learning," in *Machine Learning and Knowledge Discovery in Databases*, pp. 31–46, Springer, 2009.

[75] Lowe, D., "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[76] Lu, H., Setiono, R., and Liu, H., "Effective data mining using neural networks," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 8, no. 6, pp. 957–961, 1996.

[77] Marti, P. and Giusti, L., "A robot companion for inclusive games: A user-centred design perspective," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pp. 4348 –4353, May 2010.

[78] Martin, J.-P. and Oebel, G., "Learning by teaching: a paradigm shift in teaching," *German lessons in Japan, Journal of the Japanese Association of Teachers*, no. 12, p. 12, 2007.

[79] Matarić, M., Eriksson, J., Feil-Seifer, D., and Winstein, C., "Journal of neuroengineering and rehabilitation," *Journal of NeuroEngineering and Rehabilitation*, vol. 4, p. 5, 2007.

[80] McClanahan, B., Williams, K., Kennedy, E., and Tate, S., "A breakthrough for josh: How use of an ipad facilitated reading improvement," *TechTrends*, vol. 56, no. 3, pp. 20–28, 2012.

[81] Medin, D. L., Altom, M. W., and Murphy, T. D., "Given versus induced category representations: Use of prototype and exemplar information in classification," *Journal of Experimental Psychology: Learning, Memory, and Cognition*, vol. 10, no. 3, p. 333, 1984.

[82] Michalowski, M., Sabanovic, S., and Kozima, H., "A dancing robot for rhythmic social interaction," in *Proceedings of the ACM/IEEE international conference on Human-robot interaction*, pp. 89–96, ACM, 2007.

[83] Myllymaki, P. and Tirri, H., "Bayesian case-based reasoning with neural networks," in *Neural Networks, 1993., IEEE International Conference on*, pp. 422–427, IEEE, 1993.

[84] Nagai, Y., "From bottom-up visual attention to robot action learning," in *Development and Learning, 2009. ICDL 2009. IEEE 8th International Conference on*, pp. 1–6, IEEE, 2009.

[85] Ng, A. Y., Coates, A., Diel, M., Ganapathi, V., Schulte, J., Tse, B., Berger, E., and Liang, E., "Autonomous inverted helicopter flight via reinforcement learning," in *Experimental Robotics IX*, pp. 363–372, Springer, 2006.

[86] Nicolescu, M. N. and Mataric, M. J., "Learning and interacting in human-robot domains," *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, vol. 31, no. 5, pp. 419–430, 2001.

[87] Ontanón, S., Bonnette, K., Mahindrakar, P., Gómez-Martín, M. A., Long, K., Radhakrishnan, J., Shah, R., and Ram, A., "Learning from human demonstrations for real-time case-based planning," in *Artificial Intelligence (IJCAI) Workshop on Learning Structural Knowledge From Observations, The 2009 International Joint Conferences on*, Citeseer, 2009.

[88] Ontañón, S., Mishra, K., Sugandh, N., and Ram, A., "Learning from demonstration and case-based planning for real-time strategy games," in *Soft Computing Applications in Industry*, pp. 293–310, Springer, 2008.

[89] Ontanón, S., Montana, J. L., and Gonzalez, A. J., "Towards a unified framework for learning from observation," in *Artificial Intelligence (IJCAI) Workshop on Agents Learning Interactively from Human Teachers, The 2011 International Joint Conferences on*, 2011.

[90] Park, C. H. and Howard, A. M., "Towards real-time haptic exploration using a mobile robot as mediator," in *Haptics Symposium, 2010 IEEE*, pp. 289–292, IEEE, 2010.

[91] Park, H. W., Coogle, R., and Howard, A., "Using a shared tablet workspace for interactive demonstrations during human-robot learning scenarios," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pp. 207–208, IEEE Press, 2014.

[92] Park, H. W., Howard, A., and Kemp, C., "Understanding a child's play for robot interaction by sequencing play primitives using hidden markov models," in *Robotics and Automation, 2010. ICRA '10. IEEE International Conference on*, May 2010.

[93] Park, H. W. and Howard, A., "Providing tablets as collaborative-task workspace for human-robot interaction," in *Proceedings of the 8th ACM/IEEE international conference on Human-robot interaction*, pp. 207–208, IEEE Press, 2013.

[94] Piaget, J., *Play, dreams and imitation in childhood*. New York: Norton, 1962.

[95] Policastro, C. A., Carvalho, A. C., and Delbem, A. C., "A hybrid case adaptation approach for case-based reasoning," *Applied Intelligence*, vol. 28, no. 2, pp. 101–119, 2008.

[96] Rabiner, L., "A tutorial on hidden markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, pp. 257–286, Feb. 1989.

[97] Recio-García, J. A. and Díaz-Agudo, B., "Ontology based cbr with jcolibri," in *Applications and Innovations in Intelligent Systems XIV*, pp. 149–162, Springer, 2007.

[98] Reed, R., "Pruning algorithms-a survey," *Neural Networks, IEEE Transactions on*, vol. 4, no. 5, pp. 740–747, 1993.

[99] Robins, B., Dautenhahn, K., and Dickerson, P., "From isolation to communication: a case study evaluation of robot assisted play for children with autism with a minimally expressive humanoid robot," *Proc. Second Inter. Conf. Advances in CHI, ACHI'09.*, 2009.

[100] Roncancio, C., Rodrguez, J. L., Zalama, E., Gmez, G.-B., and others, "Improvement in service robot's interaction through case based reasoning," in *Neural Networks (IJCNN), The 2010 International Joint Conference on*, pp. 1–7, IEEE, 2010.

[101] Ros, R., Arcos, J. L., Lopez de Mantaras, R., and Veloso, M., "A case-based approach for coordinated action selection in robot soccer," *Artificial Intelligence*, vol. 173, no. 9, pp. 1014–1039, 2009.

[102] Rosch, E., "Principles of categorization," *Concepts: core readings*, pp. 189–206, 1999.

[103] Sarle, W. S., *Neural networks and statistical models*. Citeseer, 1994.

[104] Scardi, M. and Harding Jr, L. W., "Developing an empirical model of phytoplankton primary production: a neural network case study," *Ecological modelling*, vol. 120, no. 2, pp. 213–223, 1999.

[105] Scassellati, B., "How social robots will help us to diagnose, treat, and understand autism," in *Proc. of the 12th International Symposium of Robotics Research (ISSR05)*, 2005.

[106] Schaal, S., Atkeson, C. G., and Vijayakumar, S., "Scalable techniques from nonparametric statistics for real time robot learning," *Applied Intelligence*, vol. 17, no. 1, pp. 49–60, 2002.

[107] Setiono, R., "Extracting rules from pruned neural networks for breast cancer diagnosis," *Artificial Intelligence in Medicine*, vol. 8, no. 1, pp. 37–51, 1996.

[108] Setiono, R. and Liu, H., "Neural-network feature selector," *Neural Networks, IEEE Transactions on*, vol. 8, no. 3, pp. 654–662, 1997.

[109] Shah, N., "Special education pupils find learning tool in ipad applications," *Education Week*, vol. 30, no. 22, pp. 1–16, 2011.

[110] Shick, A., "Romibo robot project: an open-source effort to develop a low-cost sensory adaptable robot for special needs therapy and education," in *ACM SIGGRAPH 2013 Studio Talks*, p. 16, ACM, 2013.

[111] Shin, C.-K., Yun, U. T., Kim, H. K., and Park, S.-C., "A hybrid approach of neural network and memory-based learning to data mining," *Neural Networks, IEEE Transactions on*, vol. 11, pp. 637–646, May 2000.

[112] Smyth, B. and Cunningham, P., "Complexity of adaptation in real-world case-based reasoning systems," in *6th Irish Conference on Artificial Intelligence & Cognitive Science*, Trinity College Dublin, Department of Computer Science, 1993.

[113] Specht, D. F., "A general regression neural network," *Neural Networks, IEEE Transactions on*, vol. 2, no. 6, pp. 568–576, 1991.

[114] Starner, T. and Pentland, A., "Visual recognition of american sign language using hidden Markov models.," in *International Workshop on Automatic Face and Gesture Recognition*, pp. 2982–2987, 1995.

[115] Tapus, A., Ţăpuş, C., and Matarić, M. J., "User-robot personality matching and assistive robot behavior adaptation for post-stroke rehabilitation therapy," *Intelligent Service Robotics*, vol. 1, no. 2, pp. 169–183, 2008.

[116] Thomaz, A. L. and Breazeal, C., "Transparency and socially guided machine learning," in *5th Intl. Conf. on Development and Learning (ICDL)*, 2006.

[117] Thomaz, A. L., Breazeal, C., Barto, A. G., and Picard, R., "Socially guided machine learning," *Computer Science Department Faculty Publication Series*, p. 183, 2006.

[118] Towell, G. G. and Shavlik, J. W., "Refining symbolic knowledge using neural networks," *Machine Learning: A Multistrategy Approach*, vol. 4, pp. 405–429, 1994.

[119] Trevor, A., Park, H., Howard, A., and Kemp, C., "Playing with toys: towards autonomous robot manipulation for therapeutic play," in *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pp. 2139–2145, IEEE, 2009.

[120] Veloso, M., Carbonell, J., Perez, A., Borrajo, D., Fink, E., and Blythe, J., "Integrating planning and learning: The prodigy architecture," *Journal of Experimental & Theoretical Artificial Intelligence*, vol. 7, no. 1, pp. 81–120, 1995.

[121] Wada, K., Shibata, T., Saito, T., Sakamoto, K., and Tanie, K., "Psychological and social effects of one year robot assisted activity on elderly people at a health service facility for the aged," in *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pp. 2785–2790, IEEE, 2005.

[122] Wainer, J., Feil-Seifer, D. J., Shell, D. A., and Mataric, M. J., "The role of physical embodiment in human-robot interaction," in *Robot and Human Interactive Communication, 2006. ROMAN 2006. The 15th IEEE International Symposium on*, pp. 117–122, IEEE, 2006.

[123] Ware, M., Frank, E., Holmes, G., Hall, M., and Witten, I. H., "Interactive machine learning: letting users build classifiers," *International Journal of Human-Computer Studies*, vol. 55, no. 3, pp. 281–292, 2001.

[124] Watson, I., *Applying case-based reasoning: techniques for enterprise systems*. Morgan Kaufmann Publishers Inc., 1998.

[125] Weigend, A. S., Rumelhart, D. E., and Huberman, B. A., "Generalization by weight-elimination applied to currency exchange rate prediction," in *Neural Networks, 1991, IJCNN-91-Seattle International Joint Conference on*, vol. 1, pp. 837–841, IEEE, 1991.

[126] Wettschereck, D., Aha, D. W., and Mohri, T., "A review and empirical evaluation of feature weighting methods for a class of lazy learning algorithms," *Artificial Intelligence Review*, vol. 11, no. 1-5, pp. 273–314, 1997.

[127] Wettschereck, D. and Aha, D., "Weighting features," in *Case-Based Reasoning Research and Development* (Veloso, M. and Aamodt, A., eds.), vol. 1010 of *Lecture Notes in Computer Science*, pp. 347–358, Springer Berlin Heidelberg, 1995.

[128] Wilke, W., Vollrath, I., Althoff, K.-D., and Bergmann, R., "A framework for learning adaptation knowledge based on knowledge light approaches," in *Proceedings of the Fifth German Workshop on Case-Based Reasoning*, pp. 235–242, 1997.

[129] Wiratunga, N., Craw, S., and Rowe, R., "Learning to adapt for case-based design," in *Advances in Case-Based Reasoning*, pp. 421–435, Springer, 2002.

[130] Yamato, J., Ohya, J., and Ishii, K., "Recognizing human action in time-sequential images using hidden markov model," in *Computer Vision and Pattern Recognition, 1992. Proceedings CVPR '92., 1992 IEEE Computer Society Conference on*, pp. 379–385, June 1992.

[131] Yegnanarayana, B., *Artificial neural networks*. PHI Learning Pvt. Ltd., 2009.

[132] Zang, P., Tian, R., Thomaz, A. L., and Isbell, C. L., "Batch versus interactive learning by demonstration," in *Development and Learning (ICDL), 2010 IEEE 9th International Conference on*, pp. 219–224, IEEE, 2010.